



ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΥΠΟΛΟΓΙΣΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

**Ταξινόμηση Μουσικών Ειδών και Συστήματα Συστάσεων
με χρήση Βαθιάς Μάθησης**

ΣΠΑΝΟΥΔΑΚΗΣ ΚΩΝΣΤΑΝΤΙΝΟΣ

A.M. 71346787

Επιβλέπουσα Καθηγήτρια: Παναγιώτα Τσελέντη, Ε.ΔΙ.Π.

ΑΙΓΑΛΕΩ, Φεβρουάριος 2024

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ταξινόμηση Μουσικών Ειδών και Συστήματα Συστάσεων
με χρήση Βαθιάς Μάθησης

Music Genre Classification and Recommender Systems
using Deep Learning

Σπανουδάκης Κωνσταντίνος

A.M. 71364787

Γλώσσα Συγγραφής: Ελληνικά

Μέλη Εξεταστικής Επιτροπής συμπεριλαμβανομένου και του Εισηγητή

A/A	ΟΝΟΜΑΤΕΠΩΝΥΜΟ	ΒΑΘΜΙΔΑ/ΙΔΙΟΤΗΤΑ	ΥΠΟΓΡΑΦΗ
1	ΤΣΕΛΕΝΤΗ ΠΑΝΑΓΙΩΤΑ	ΕΔΙΠ	
2	ΧΡΗΣΤΟΣ ΤΡΟΥΣΣΑΣ	ΕΠΙΚΟΥΡΟΣ ΚΑΘΗΓΗΤΗΣ	
3	ΑΚΡΙΒΗ ΚΡΟΥΣΚΑ	ΜΕΤΑΔΙΔΑΚΤΟΡΙΚΗ ΕΡΕΥΝΗΤΡΙΑ	

Ημερομηνία Εξέτασης 19 Μαρτίου 2024

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος Σπανουδάκης Κωνσταντίνος του Νικολάου, με αριθμό μητρώου 71346787 φοιτητής του Τμήματος Μηχανικών Πληροφορικής και Υπολογιστών της Σχολής Μηχανικών του Πανεπιστημίου Δυτικής Αττικής, δηλώνω ότι:

«Βεβαιώνω ότι είμαι συγγραφέας αυτής της Διπλωματικής εργασίας και κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματός.

Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

Ο Δηλών



ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω την επιβλέπουσα καθηγήτρια κυρία Παναγιώτα Τσελέντη για την εμπιστοσύνη της και την καθοδήγηση που μου προσέφερε ανοίγοντας μου τον δρόμο σε έναν πολύ ενδιαφέρον τομέα της Επιστήμης των Υπολογιστών τον οποίο εξερεύνησα μέσα από αυτή την εργασία.

Επίσης, θέλω να εκφράσω την ευγνωμοσύνη μου προς τους κοντινούς και αγαπημένους ανθρώπους - γονείς, δασκάλους και φίλους μου - για την στήριξη, την καθοδήγηση και την πίστη που μου παρέχουν.

ΠΕΡΙΛΗΨΗ

Στο σημερινό δυναμικό μουσικό οικοσύστημα, η σημασία της ταξινόμησης μουσικών ειδών (MGC) και των συστημάτων μουσικών συστάσεων (MRS) δεν μπορεί να υπερεκτιμηθεί. Με τον πολλαπλασιασμό των ψηφιακών μουσικών πλατφορμών και την άνευ προηγουμένου αφθονία περιεχομένου, η MGC χρησιμεύει ως άξονας για τη διαισθητική οργάνωση και πλοήγηση σε εκτεταμένες μουσικές βιβλιοθήκες. Ταυτόχρονα, το MRS διαδραματίζει κρίσιμο ρόλο στη βελτίωση της εμπειρίας του χρήστη, παρέχοντας εξατομικευμένες συστάσεις μπροστά στις ποικίλες και εξελισσόμενες προτιμήσεις των ακροατών. Η σημασία του MGC και του MRS εκτείνεται πέρα από την απλή ευκολία, διαμορφώνοντας τον τρόπο με τον οποίο οι χρήστες εξερευνούν και συνδέονται με τη μουσική με εξατομικευμένο και ουσιαστικό τρόπο. Τα συστήματα αυτά όχι μόνο απλοποιούν την πρόσβαση σε ένα διαρκώς διευρυνόμενο μουσικό τοπίο, αλλά συμβάλλουν επίσης στον πολιτιστικό ιστό της ψηφιακής κοινωνίας μας, προωθώντας την ανακάλυψη, την εμπλοκή και την απόλαυση στο πεδίο της κατανάλωσης μουσικής.

Η παρούσα διπλωματική εργασία ασχολείται με την πρακτική εφαρμογή μεθοδολογιών βαθιάς μάθησης για την ταξινόμηση μουσικών ειδών (MGC) και για συστήματα μουσικών συστάσεων (MRS). Επικεντρώνοντας αποκλειστικά στη βαθιά μάθηση, η μελέτη διερευνά την αποτελεσματικότητα των νευρωνικών δικτύων χρησιμοποιώντας Multilayer Perceptron (MLP), Convolutional Neural Networks (CNN) και Long-Short Term Memory Network (LSTM) για την ταξινόμηση μουσικών ειδών και μια προσέγγιση βασισμένη στο περιεχόμενο (content-based approach) για τα συστήματα συστάσεων. Κύρια μορφή δεδομένων και πληροφορίας η οποία θα μας δώσει την δυνατότητα να εξάγουμε πληροφορία και συμπεράσματα για το κάθε μουσικό κομμάτι θα είναι τα Mel-Frequency Cepstral Coefficients (MFCC).

Επιπλέον πραγματοποιούμε μια Βιβλιογραφική Ανασκόπηση που παρέχει μια επισκόπηση της υπάρχουσας έρευνας στην MGC πλαίσιο της βαθιάς μάθησης και τις content-based προσεγγίσεις που υπάρχουν στον τομέα του MRS, αναφερόμαστε στο τι υπάρχει στον κόσμο του Art & AI σήμερα και πιο συγκεκριμένα εφαρμογές που αφορούν την χρήση της Τεχνητής Νοημοσύνης στην Μουσική. Επίσης αναλύουμε τις βασικές γνώσεις που χρειάζονται για την κατανόηση όλων των εργαλείων που χρησιμοποιούμε στην παρούσα εργασία όπως για παράδειγμα πως λειτουργεί ένα CNN, τι είναι τα MFCC κι άλλα. Τέλος υλοποιούμε ένα Web App στο οποία αναδεικνύεται η λειτουργικότητα του μοντέλου που δημιουργήσαμε και το θέτουμε σε εφαρμογή με γνωστά και άγνωστα σε αυτό μουσικά κομμάτια.

ΕΠΙΣΤΗΜΟΝΙΚΗ ΠΕΡΙΟΧΗ: Τεχνητή Νοημοσύνη

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ, ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ, ΜΟΥΣΙΚΗ, ΒΑΘΙΑ ΜΑΘΗΣΗ, ΑΝΑΚΤΗΣΗ ΜΟΥΣΙΚΗΣ ΠΛΗΡΟΦΟΡΙΑΣ

ABSTRACT

In today's dynamic musical ecosystem, the importance of music genre classification (MGC) and music recommendation systems (MRS) cannot be overestimated. With the proliferation of digital music platforms and the unprecedented abundance of content, MGC serves as a linchpin for intuitive organization and navigation of extensive music libraries. At the same time, MRS plays a critical role in enhancing the user experience by providing personalized recommendations in the face of listeners' diverse and evolving preferences. The importance of MGC and MRS extends beyond mere convenience, shaping the way users explore and connect with music in a personalized and meaningful way. These systems not only simplify access to an ever-expanding musical landscape, but also contribute to the cultural fabric of our digital society by promoting discovery, engagement and enjoyment in the realm of music consumption.

This thesis deals with the practical application of deep learning methodologies for music genre classification (MGC) and music recommendation systems (MRS). Focusing exclusively on deep learning, the study investigates the effectiveness of neural networks using Multilayer Perceptron (MLP), Convolutional Neural Networks (CNN), and Long-Short Term Memory (LSTM) for music genre classification and a content-based approach for recommendation systems. The main form of data and information that will enable us to extract information and inferences about each musical piece will be Mel-Frequency Cepstral Coefficients (MFCC).

In addition we conduct a Literature Review that provides an overview of existing research in the MGC framework of deep learning and the content-based approaches that exist in the field of MRS, we report on what exists in the world of Art& AI today and more specifically applications involving the use of AI in Music. We also discuss the basic knowledge needed to understand all the tools used in this work, such as how a CNN works, what MFCCs are and more. Finally, we implement a Web App in which we demonstrate the functionality of the model we created and put it into practice with known and unknown music tracks.

SCIENTIFIC AREA: Artificial Intelligence

KEYWORDS: ARTIFICIAL INTELLIGENCE, MACHINE LEARNING, MUSIC, DEEP LEARNING, MUSIC INFORMATION RETRIEVAL

Περιεχόμενα

1	Εισαγωγή	16
1.1	Ερευνητικοί Στόχοι	17
1.2	Δομή Εργασίας	17
1.3	Τέχνη και ΑΙ	19
1.3.1	Ιστορία του ΑΙ στην Τέχνη	20
1.3.2	ΑΙ και Τέχνη σήμερα	22
1.3.3	ΑΙ και Μουσική σήμερα	24
1.4	Music Information Retrieval	29
2	Βιβλιογραφική Ανασκόπηση	30
2.1	Music Genre Classification	30
2.2	Recommender Systems	34
2.2.1	Context-Based	34
2.2.2	Collaborative Filtering	35
2.2.3	Content-Based	35
3	Deep Learning	38
3.1	Ο Νευρώνας	39
3.2	Συναρτήσεις Ενεργοποίησης	40
3.3	Multilayer Perceptron - Forward Propagation	41
3.4	Back Propagation	45
3.4.1	Gradient Descent	47
3.5	Είδη Gradients	49
3.5.1	Vanishing Gradients	50
3.6	Το Πρόβλημα της Υπερπροσαρμογής	51
3.7	Convolutional Neural Networks	55
3.7.1	Convolution	56
3.7.2	Pooling	57
3.7.3	Μετάβαση σε Fully-Connected Layer	57
3.8	Recurrent Neural Networks	58
3.8.1	Αρχιτεκτονική	59

3.8.2	LSTM	61
4	Ήχος & Μουσική	64
4.1	Ήχος	64
4.2	Μετρικές του Ήχου	66
4.3	Fourier Transform	66
4.3.1	Short Time Fourier Transform	68
4.4	Mel Frequency Cepstral Coefficients - MFCCs	70
5	Μετρικές για Συστήματα Συστάσεων	72
5.1	Cosine Similarity	72
5.2	Euclidean Distance	73
6	Υλοποίηση & Μεθοδολογία	74
6.1	Εργαλεία	74
6.2	Προετοιμασία των Δεδομένων	75
6.3	Μετρικές	78
6.4	Μοντέλα Βαθιάς Μάθησης	79
6.4.1	Multilayer Perceptron Network	79
6.4.2	Convolutional Neural Network	81
6.4.3	Recurrent Neural Network - LSTM	84
6.5	Μηχανή δημιουργίας Συστάσεων	87
6.5.1	Με χρήση MFCC	87
6.5.2	Με χρήση άλλων Χαρακτηριστικών	88
6.5.3	Με προσθήκη Metadata	89
7	Πειραματικά Αποτελέσματα	90
7.1	Ταξινόμηση κομματιού σε Είδος	90
7.2	Σύσταση παρόμοιων κομματιών	92
8	Εκτέλεση της Τελικής Εφαρμογής	96
8.1	Εισαγωγή Κομματιών από το Dataset	96
8.2	Εισαγωγή Κομματιών εκτός του Dataset	98
9	Συμπεράσματα & Μελλοντικές Επεκτάσεις	101

10 Παράρτημα	103
10.1 Τελικός παραδοτέος Κώδικας	103
10.2 Λοιποί Κώδικες	111

Κατάλογος Σχημάτων

1	Χρήση μουσικής ανά ηλικία, έρευνα στις Η.Π.Α. 2019	16
2	Εικόνες του AARON	20
3	Η πρώτη αλγοριθμική σύνθεση που δημιουργήθηκε από υπολογιστή: το πρώτο μέρος από το Iliac Suite για κουαρτέτο εγχόρδων	21
4	Μαθηματική αναπαράσταση του Pithoprakta, ενός πρώιμου έργου του Ξενάκη. Μοντελοποιεί τους διαφορετικούς ηχητικούς χώρους μέσω αυτών των Glissandi (το γλίστρημα από το ένα τονικό ύψος στο άλλο) ως γραφήματα ευθειών γραμμών (ο χρόνος στον οριζόντιο άξονα, το τονικό ύψος στον κάθετο), όπου διαφορετικές κλίσεις αντιστοιχούν σε διαφορετικούς "ηχητικούς χώρους".	21
5	Παράδειγμα του Dall-e 2. Πάνω φαίνεται το text prompt με αυτό που ζητήθηκε να δημιουργήσει	22
6	Δεύτερο παράδειγμα.	23
7	Théâtre D'opéra Spatial, μια εικόνα του Midjourney που κέρδισε το πρώτο βραβείο σε διαγωνισμό ψηφιακής τέχνης.	23
8	Μια ψεύτικη εικόνα του Πάπα Φραγκίσκου που δημιούργησε το Midjourney, η οποία έγινε viral το 2023.	24
9	Δημιουργία ποιήματος βάση περιγραφής	24
10	Pedalboard Κιθάρας	27
11	Music Tagging παράδειγμα	27
12	Το μοντέλο MusicLM	28
13	Spectrograms για τα Music Genre της έρευνας [1]	31
14	Αποτελέσματα της έρευνας [2]	33
15	Αρχιτεκτονική του Late Fusion μοντέλου από το [2]	34
16	Πίνακας παρμένος από το [3]	36
17	Σύγκριση Traditional ML με DL	39
18	Ο Νευρώνας Perceptron	39
19	Sigmoid Καμπύλη	40
20	ReLU Καμπύλη	41
21	Βαθύ Νευρωνικό Δίκτυο	42
22	MultiLayer Perceptron	42
23	Ενεργοποιημένοι νευρώνες και βάρη για τον υπολογισμό του $h_1^{(2)}$	44
24	Βαθύ Νευρωνικό Δίκτυο με 1 Hidden Layer	46
25	Gradient Descent	47
26	Η σιγμοειδής συνάρτηση και η παράγωγος της	51
27	Overfitting Error (https://quantdare.com/mitigatingoverfittingneuralnetworks/)	52
28	Early Stop	53
29	Dropout Method (https://medium.com)	54
30	Αρχιτεκτονική ενός CNN https://www.linkedin.com/in/nourhan-moustafa/	55
31	Η πράξη της Συνέλιξης https://www.researchgate.net/	56
32	Διάφοροι τύποι Kernels https://medium.com/analytics-vidhya/	57
33	Max και Average Pooling https://towardsai.net/p/v/	58

34	Αρχιτεκτονική ενός RNN https://github.com/musikalkemist	59
35	Ο πυρήνας του RNN https://github.com/musikalkemist	59
36	Ο πιο απλός πυρήνας RNN https://medium.com/@imjeremyhi/	60
37	Χάσιμο παλιάς πληροφορίας από RNN https://towardsdatascience.com/	60
38	Πυρήνας του LSTM https://www.linkedin.com/pulse	61
39	Άλλη αναπαράσταση του LSTM https://thorirmar.com/post/insight_into_lstm/	61
40	Σύγκριση RNN (πάνω) με LSTM (κάτω)	62
41	Waveform	64
42	Μεγάλο και Μικρό Πλάτος	65
43	Μεγάλη και Μικρή Συχνότητα	65
44	Μετατροπή από Analog σε Digital	66
45	Soundwave ενός πλήκτρο από Πιάνο	67
46	Μετασχηματισμός Fourier	67
47	Μετασχηματισμός Fourier	68
48	Spectrogram του STFT	69
49	Spectrogram του STFT	69
50	Αναπαράσταση του MFCC	71
51	Γραφικό παράδειγμα cosine similarity	73
52	Κατάλογος τραγουδιών	75
53	Εικόνες των Κομματιών	76
54	Αρχιτεκτονική του MultiLayer Perceptron Network	80
55	Εκπαίδευση του μοντέλο ανά Epoch	81
56	Error & Accuracy μετά από Regularization	82
57	Η αρχιτεκτονική του CNN μοντέλου	84
58	Units σε ένα LSTM https://towardsdatascience.com/	85
59	Sequence to Vector LSTM	85
60	Sequence to Sequence LSTM	85
61	Τελικό LSTM μοντέλο	86
62	Multilayer Perceptron Network	90
63	Convolutional Neural Network	91
64	Long Short Term Memory Network	91
65	Home Page της Εφαρμογής	96
66	Κομμάτι: rock.00089.wav	96
67	Κομμάτι: metal.00065.wav	97
68	Κομμάτι: disco.00031.wav	97
69	Κομμάτι: jazz.00024.wav	97
70	Κομμάτι: classical.00014.wav	98
71	Κομμάτι: Earth Wind & Fire – Boogie Wonderland	98
72	Κομμάτι: The Doors – Roadhouse Blues	99
73	Κομμάτι: Metallica – Seek and Destroy	99
74	Κομμάτι: Mozart – Lacrimosa	100

Κατάλογος Πινάκων

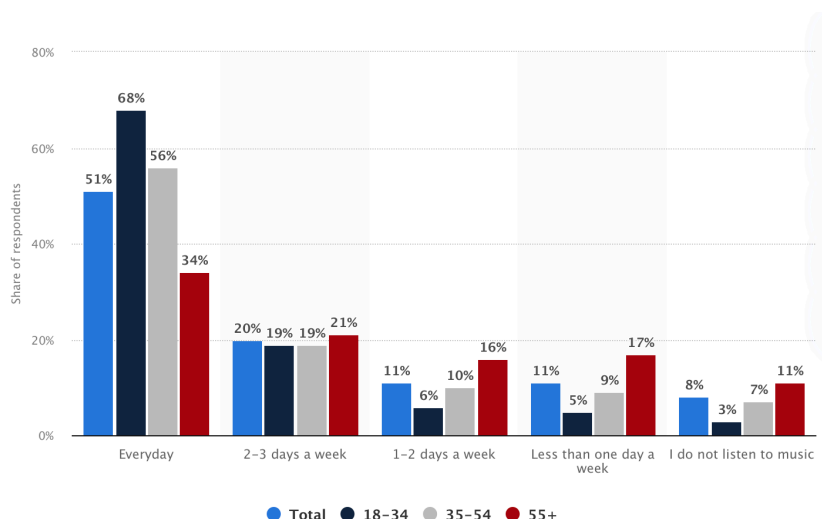
1	Παράμετροι του preprocess	77
2	Χαρακτηριστικά που θα εξαχθούν	88
3	Αποτελέσματα των Μοντέλων	90

Κατάλογος Συντομογραφιών

- AI** Artificial Intelligence
- AL** Adaptive Learning
- AMGC** Automatic Music Genre Classification
- DIP** Digital Image Processing
- DL** Deep Learning
- FFT** Fast Fourier Transform
- IR** Information Retrieval
- LSTM** Long-Short Term Memory
- MGC** Music Genre Classification
- MIR** Music Information Retrieval
- ML** Machine Learning
- MLP** MultiLayer Perceptron
- MFCC** Mel Frequency Cepstral Coefficients
- MRC** Music Recommender Systems
- NN** Neural Networks
- NLP** Natural Language Processing
- RNN** Recurrent Neural Networks
- RS** Recommender System
- STFT** Short Time Fourier Transform

1 Εισαγωγή

Η μουσική αποτελεί ένα αναπόσπαστο κομμάτι της καθημερινότητας για την πλειοψηφία των ανθρώπων, σε μια έρευνα του 2003 [4] δείχνει ότι ακόμη και 20 χρόνια πριν οι άνθρωποι έτειναν να ακούν περισσότερο μουσική από το να ψυχαγωγούνται με οποιοδήποτε άλλο μέσο όπως μεταξύ άλλων τηλεόραση, διάβασμα και παρακολούθηση κάποιας ταινίας. Με τον ερχομό και την εξέλιξη εφαρμογών όπως Spotify, Deezer, Soundcloud κι άλλων δόθηκε μεγάλη ώθηση σε καλλιτέχνες χωρίς μεγάλα στούντιο, χωρίς ακριβό εξοπλισμό και με περιορισμένους πόρους να δημιουργούν μουσική, να την προωθούν και να βρίσκουν αντίκρισμα [5]. Στην Εικόνα 1 φαίνεται πως άτομα μικρότερων ηλικιών είναι αυτά που ακούν περισσότερο μουσική στην καθημερινότητα τους με την πλειοψηφία να ακούει κάθε ημέρα, αυτό μας δείχνει ότι με την πάροδο των χρόνων οι άνθρωποι τείνουν να ακούν όλο και περισσότερο μουσική. Το Spotify ανάρτησε στην ιστοσελίδα του την αύξηση των Premium συνδρομητών του ανά τα χρόνια ξεκινώντας από 18 εκατομμύρια το 2015 και φτάνοντας στα 226 εκατομμύρια το 2023¹. Ένας από τους πιο βασικούς παράγοντες των χρηστών για την προτίμηση τους για την εφαρμογή από την οποία θα ακούν μουσική είναι η ικανότητα της εκάστοτε εφαρμογής να τους προτείνει κομμάτια ανάλογα με αυτά που ακούν [6].



Εικόνα 1: Χρήση μουσικής ανά ηλικία, έρευνα στις Η.Π.Α. 2019

Ως αποτέλεσμα των λίγων στοιχείων που αναφέραμε παραπάνω γεννιέται απευθείας η ανάγκη για συστήματα τα οποία ενεργούν στον τομέα της μουσικής με διάφορους τρόπους οι οποίοι καθιστούν αυτό το γιγάντιο πλήθος μουσικών τραγουδιών προσβάσιμο στους χρήστες, στο υποκεφάλαιο 1.3.3 αναφερόμαστε πολλούς τομείς στους οποίους επιδρά η τεχνητή νοημοσύνη πάνω στην μουσική. Σήμερα, όπου η αφθονία της μουσικής μπορεί να είναι ταυτόχρονα ευλογία και πρόκληση και υπάρχουν έρευνες που ασχολούνται με το αν η μουσική πλέον είναι ένα υγιές χόμπι ή μια εξάρτηση για την καθημερινότητα του ανθρώπου [7], ο ρόλος της τεχνολογίας στην καθοδήγηση των ακροατών μέσα σε αυτό το τεράστιο ηχητικό τοπίο έχει γίνει όλο και πιο σημαντικός. Τα συστήματα μουσικών συστάσεων, τα οποία

¹<https://www.demandsage.com/spotifystats/>

βασίζονται σε εξελιγμένους αλγόριθμους και μηχανική μάθηση, προσφέρουν στους χρήστες εξατομικευμένα μονοπάτια για να ανακαλύψουν νέα κομμάτια, ενώ η λεπτή εργασία της ταξινόμησης των μουσικών ειδών αποκαλύπτει τα περίπλοκα στρώματα που καθορίζουν την ηχητική ταυτότητα ενός τραγουδιού και βοηθούν κομμάτια παρόμοιου είδους να ταξινομηθούν στο ανάλογο είδος τους ώστε να δημιουργηθεί μια δεξαμενή παρεμφερή κομματιών με αυτό που αναζητά ο χρήστης.

1.1 Ερευνητικοί Στόχοι

Ο κύριος στόχος της εργασίας είναι να δημιουργηθεί μια εφαρμογή η οποία δέχεται ένα μουσικό κομμάτι και από αυτό εξάγει το είδος μουσικής στο οποίο ανήκει και να προτείνει παρόμοια κομμάτια με βάση τα χαρακτηριστικά του κομματιού. Άλλοι στόχοι από τους οποίους απαρτίζεται η εργασία είναι να ερευνήσουμε πάνω στις προσεγγίσεις βαθιάς μάθησης που ασχολούνται με το παραπάνω πρόβλημα και να υλοποιήσουμε τα μοντέλα νευρωνικών δικτύων που σύμφωνα με έρευνες επιλύουν βέλτιστα το θέμα το Music Genre Classification. Έπειτα να δούμε πως λειτουργούν οι κύριοι αλγόριθμοι για Music Recommender Systems βασισμένοι στα χαρακτηριστικά του ηχητικού κομματιού (*Content-Based Approaches*) και να υλοποιήσουμε έναν τέτοιο. Ακόμη ένας στόχος είναι να δουλέψουμε με την βιβλιοθήκη Tensorflow η οποία είναι από τις πιο δημοφιλείς βιβλιοθήκες Βαθιάς Μάθησης σε Python που χρησιμοποιείται κατά κόρον στην βιομηχανία. Τέλος είναι να δώσουμε μια εικόνα του τι υπάρχει στον τομέα της Επιστήμης των Υπολογιστών που ασχολείται με την εξαγωγή πληροφορίας από μουσικό κομμάτι ονόματι Music Information Retrieval, και σε τι στάδιο βρίσκεται η έρευνα πάνω σε αυτόν τον τομέα σήμερα.

1.2 Δομή Εργασίας

Η εργασία θα διαιρεθεί τους παρακάτω τομείς.

1. *Βιβλιογραφική Ανασκόπηση*. Αρχικά θα ερευνήσουμε τις διάφορες προσεγγίσεις και αλγόριθμους που υπάρχουν πάνω στους τομείς του MGC και MRS. Εδώ θα αναφέρουμε με ποια μέθοδο θα εργαστούμε για την δική μας υλοποίηση. Θα κάνουμε μια ανασκόπηση πάνω στις γνωστές και πετυχημένες εφαρμογές που υπάρχουν στον τομέα της Μουσικής με χρήση AI και θα δούμε συνοπτικά πως ιστορικά αυτές εξελίχθηκαν.
2. *Βαθιά Μάθηση*. Εδώ θα αναλύσουμε τον τρόπο που δουλεύουν τα νευρωνικά δίκτυα παρουσιάζοντας 2 μοντέλα, το Multilayer Perceptron και το Convolutional Neural Network. Εδώ θέτουμε τα μαθηματικά θεμέλια που θα μας οδηγήσουν στην κατανόηση των μοντέλων που θα χρησιμοποιήσουμε στην εφαρμογή.
3. *Το Ήχος & Μουσική*. Αναλύουμε την μορφή των δεδομένων που θα εισαχθούν στα μοντέλα βαθιάς μάθησης που θα δημιουργήσουμε. Θα αναφερθούμε στα βασικά χαρακτηριστικά που εξάγονται από τον ήχο και πως αυτά χρησιμοποιούνται για να εκπαιδευτούν τα νευρωνικά δίκτυα.

4. *Recommender Systems*. Σε αυτό το κεφάλαιο θα σκιαγραφήσουμε την τεχνική που θα εφαρμόσουμε για να προτείνει το σύστημα μας παρόμοια τραγούδια που αυτό που ει-σήχθη.
5. *Υλοποίηση*. Σε αυτό το σημείο θα δείξουμε πως θα δημιουργήσουμε το μοντέλο βάση των προδιαγραφών που θεσπίσαμε στα προηγούμενα κεφάλαια. Αναφερόμαστε σε θέματα που μας δυσκόλεψαν και τις λύσεις που εφαρμόσαμε για να τα υπερβούμε. Παρουσιάζουμε τις ρυθμίσεις που θα εφαρμοστούν στα DL μοντέλα όπως και στο RS.
6. *Αποτελέσματα και Σχολιασμός*. Εδώ γίνεται μια σύγκριση των αποτελεσμάτων ανάλογα με την μέθοδο που χρησιμοποιήσαμε καταλήγοντας στην καλύτερη.
7. *Εκτέλεση της Τελικής Εφαρμογής*. Παρουσιάζουμε την διαδικτυακή διεπαφή που δημιου-ργήσαμε που πολύ απλά δέχεται ένα μουσικό κομμάτι και εξάγει από αυτό το είδος μουσικής που ανήκει και προτείνει παρόμοια τραγούδια.

Η μουσική αποτελεί μεγάλο μέρος του διαδικτύου, και το Internet αποτελεί την πιο προ-σβάσιμη πηγή για να ερχόμαστε σε επαφή με καλλιτέχνες και μουσική. Ο ανταγωνισμός των εφαρμογών πάνω στο κομμάτι της προσβασιμότητας έχει οδηγήσει εταιρίες όπως Spotify, Apple Music, Deezer να δημιουργήσουν αλγόριθμους για να προσφέρεται στον χρήστη η πιο συναφής μουσική με αυτή που επιλέγει. Χρησιμοποιούνται διάφορες μεθοδολογίες Τεχνη-τής Νοημοσύνης ώστε να ταξινομούνται δίσκοι, καλλιτέχνες, κομμάτια και playlist βάση του είδους τους, του ρυθμού τους, του συναισθήματος που βγάζουν και πολλών άλλων μετα-βλητών που καθορίζουν το ποια μουσική ταιριάζει με ποια άλλη. Έτσι εφαρμογές όπως το Spotify λαμβάνει σαν είσοδο τα τραγούδια που ακούει ο χρήστης και του δημιουργεί λίστες με κομμάτια που τον ενδιαφέρουν.

Το βασικότερο εργαλείο στην εποχή μας που χρησιμοποιείται κατά κόρον σε οποιαδή-ποτε μεγάλη και επιτυχημένη εφαρμογή είναι το το **Artificial Intelligence (AI)** και τα διά-φορα παρακλάδια του όπως **Machine Learning (ML)**, **Neural Networks (NN)**, και **Deep Learning (DL)**. Το AI χρησιμοποιείται σε μια πληθώρα γνωστών εφαρμογών σήμερα όπως για παράδειγμα στους παρακάτω τομείς:

- **Εικονικοί Βοηθοί (NLP)**. Κατανοούν και απαντούν στα ερωτήματα των χρηστών.
Παραδείγματα Εφαρμογών: Siri, Google Assistant, Alexa
- **Ηλεκτρονικό Εμπόριο (RS)**. Αναλύουν το ιστορικό περιήγησης και αγορών των χρη-στών για να προτείνουν σχετικά προϊόντα.
Παραδείγματα Εφαρμογών: Amazon
- **Εκπαίδευση (AL)**. Αναλύουν τα δεδομένα απόδοσης των μαθητών και παρέχουν εξα-τομικευμένες μαθησιακές εμπειρίες
- **Social Media**. Χρησιμοποιούν τεχνητή νοημοσύνη για τον έλεγχο του περιεχομένου για να φιλτράρουν το ακατάλληλο περιεχόμενο.
Παραδείγματα Εφαρμογών: Facebook, Instagram

- **Autonomous Vehicles.** Επικεντρώνονται στην ανάπτυξη τεχνολογίας αυτοοδηγούμενων αυτοκινήτων
Παραδείγματα Εφαρμογών: Waymo
- **Gaming.** Εργαλεία τεχνητής νοημοσύνης για την ανάπτυξη παιχνιδιών, συμπεριλαμβανομένης της συμπεριφοράς των NPC.
Παραδείγματα Εφαρμογών: Unity Technologies

Τα παραπάνω αποτελούν παραδείγματα εφαρμογών και υλοποιήσεων ΑΙ όπου καταπιάνονται με πολλούς τομείς. Τα τελευταία χρόνια έχει γίνει μια μεγάλη στροφή και έχουν αναπτυχθεί εφαρμογές **τέχνης** με χρήση ΑΙ.

1.3 Τέχνη και ΑΙ

Η σχέση μεταξύ τέχνης, ΑΙ και ανθρώπων μοιράζονται ένα κοινό θεμέλιο στην διαδικασία της δημιουργίας. Τόσο οι καλλιτέχνες όσο και τα συστήματα τεχνητής νοημοσύνης λαμβάνουν εξωτερικά ερεθίσματα² και μέσω επεξεργασίας αυτών των ερεθισμάτων δημιουργούν αποτελέσματα.

Ένας Ζωγράφος όταν θα θέλει να δημιουργήσει ένα έργο θα λάβει ερέθισμα από την εικόνα που θα φανταστεί, θα χρησιμοποιήσει την ήδη υπάρχουσα γνώση που έχει μέσα από την εκπαίδευση που έλαβε όσο καιρό ζωγραφίζει όσο και τους κανόνες που έχουν θεσπιστεί από την Τέχνη της Ζωγραφικής, θα αντλήσει έμπνευση από τους προηγούμενους καλλιτέχνες που πέρασαν στην ιστορία αλλά επίσης και από τους δικούς του πίνακες και θα δημιουργήσει το νέο του έργο. Ένας Ζωγράφος όσο πιο πολύ εξασκείται στο να ζωγραφίζει ένα δέντρο τόσο καλύτερος και θα γίνεται, αυτό γιατί χτίζει μνήμη πάνω στην διεργασία **ζωγραφίζω ένα δέντρο** και κάθε φορά την βελτιώνει μέχρι στο τέλος στην διεργασία **ζωγραφίζω ένα δέντρο** να έχει εισάγει τους καλύτερους κανόνες και πρακτικές του. Ένας καλλιτέχνης που έχει χάσει την μνήμη του³ δεν θα έχει από που να αντλήσει την γνώση για να δημιουργήσει ένα δέντρο και θα πρέπει να ξεκινήσει από την αρχή επαναλαμβάνοντας το πιο πιθανών παρόμοια λάθη με πριν.

Εκεί είναι το κοινό σημείο μας με τους υπολογιστές, όπου ένας υπολογιστής μέσα από χιλιάδες πίνακες ζωγραφικής θα έχει τελειοποιημένη την διεργασία **ζωγραφίζω ένα δέντρο** και θα μπορεί να την επιτελέσει ορθά αφού έχει εκπαιδευτεί με ένα μεγάλο πλήθος δέντρων. Στο κομμάτι που αντλούμε πληροφορίες από παρελθόν⁴ ένας υπολογιστής με έναν άνθρωπο έχουν την ίδια δυνατότητα δημιουργίας. Ωστόσο μια διαφορά έγκειται στο ότι κάτι που βασίζεται αποκλειστικά στο παρελθόν για να δημιουργήσει κάτι τότε το αποτέλεσμα θα είναι μια σύνθεση παρελθοντικών έργων, ως αποτέλεσμα τίποτα ουσιαστικά «καινούριο» δεν δημιουργείται, ένας πρωτοποριακός Picasso που έβαλε την δική του υπογραφή στην τέχνη της ζωγραφικής βασίστηκε στην ήδη υπάρχουσα ιστορία της ζωγραφικής αλλά πρόσθεσε και το

²Οι άνθρωποι σε μορφή που αναγνωρίζουν οι αισθήσεις (όραση, ακοή, όσφρηση...) και οι Η/Υ σε μορφή Datasets

³Δηλαδή και την εκπαίδευση του

⁴Μνήμη, αποκτημένη γνώση, πίνακες προηγούμενων καλλιτεχνών, κανόνες της τέχνης, τεχνικές

δικό του νέο στοιχείο μέσα. Έτσι δημιουργείται η διαφορά στον άνθρωπο και στον υπολογιστή που είναι η διαφορά της **πρωτοπορίας**.

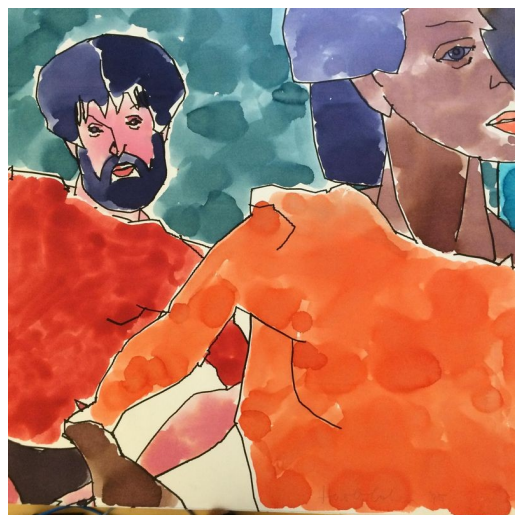
Ωστόσο κάτι τέτοιο δεν αφαιρεί από έναν Η/Υ την δυνατότητα να δημιουργεί τέχνη, παραδείγματα εφαρμογών σήμερα μας αποδεικνύουν ότι τα ΑΙ συστήματα έχουν γίνει ένα πολύ βασικό εργαλείο σε πολλούς κλάδους που το χρειάζονται ⁵ όπου μπορούν να δημιουργήσουν Εικόνες, Ποίηση, Μουσική και άλλους κλάδους της τέχνης.

1.3.1 Ιστορία του ΑΙ στην Τέχνη

Η ιστορία του ΑΙ Art ξεκινά το 1960 όπου οι επιστήμονες είχαν αρχίσει να εξερευνούν τις δυνατότητες ένας υπολογιστής να δημιουργεί τέχνη. Έτσι ένα από τα πρώτα παραδείγματα αυτής της έρευνας είναι το **AARON** του Harold Cohen ο οποίος αφιέρωσε όλη του ζωή του σε αυτό το πρόγραμμα. Το AARON χρησιμοποιούσε σετ από αλγόριθμους και κανόνες για να δημιουργεί αφηρημένες εικόνες και με το feedback ήταν ικανό να προσαρμόζεται και να μαθαίνει.



(i) Ανώνυμο 1980



(ii) Η πρώτη έγχρωμη εικόνα 1995

Εικόνα 2: Εικόνες του AARON

Στον τομέα της Μουσικής οι πρώτες απόπειρες για computer-generated music εμφανίστηκαν το 1957 [8] όπου η εστίαση ήταν πάνω στην αλγοριθμική ανάπτυξη μουσικής. Εκείνη την χρονιά βλέπουμε το πρώτο μουσικό έργο το οποίο δημιουργήθηκε αποκλειστικά από τεχνητή νοημοσύνη το *Illiac Suite for String Quartet*. Για αυτό το εγχείρημα χρησιμοποιήθηκε ένας Monte Carlo αλγόριθμος ο οποίος παρήγαγε τυχαίους αριθμούς που αντιστοιχούσαν σε ορισμένα μουσικά χαρακτηριστικά όπως το τονικό ύψος ή ο ρυθμός. Εφαρμόζοντας ένα σύνολο περιορισμών, αυτά τα αυθαίρετα χαρακτηριστικά περιορίστηκαν σε στοιχεία που τηρούν τις μουσικές συμβάσεις που θεωρούνται "νόμιμες" σύμφωνα με τους παραδοσιακούς κανόνες της μουσικής θεωρίας, τις στατιστικές πιθανότητες (όπως οι αλυσίδες Markov) και

⁵ Διαφημιστές, Content Creators, Διοργανωτές Εκδηλώσεων κ.α.

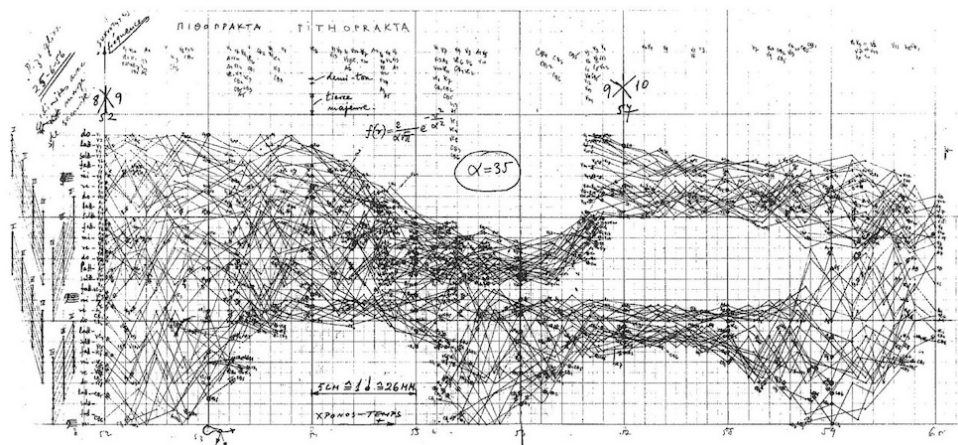
τις δημιουργικές ιδέες των δύο συνθετών.



Εικόνα 3: Η πρώτη αλγοριθμική σύνθεση που δημιουργήθηκε από υπολογιστή: το πρώτο μέρος από το Illiac Suite για κουαρτέτο εγχόρδων

Ένας ακόμη πρωτοπόρος στον τομέα είναι ο Ιάννης Ξενάκης. Χρησιμοποιώντας στοχαστικές πιθανότητες στις μουσικές του προσπάθειες, χρησιμοποίησε μια στοχαστική διαδικασία που χαρακτηρίζεται από τυχαίες κατανομές πιθανοτήτων που αφηφούν την πρόβλεψη αλλά μπορούν να αναλυθούν στατιστικά. Στις αρχές της δεκαετίας του 1960, αξιοποίησε τους υπολογιστές και τη γλώσσα FORTRAN για να διαπλέξει διάφορες συναρτήσεις πιθανοτήτων, καθορίζοντας τη συνολική δομή και τις πρόσθετες παραμέτρους (όπως το τονικό ύψος και η δυναμική) μιας σύνθεσης. Ο Ξενάκης διαμόρφωσε τη μουσική του σαν να διαμόρφωνε ένα επιστημονικό πείραμα. Κάθε όργανο ήταν σαν ένα μόριο και θα υποβαλλόταν στη δική του στοχαστική, τυχαία διαδικασία για να καθορίσει τη συμπεριφορά του (τη συχνότητα του τόνου και την ταχύτητα ορισμένων νοτών).

Figure 1: Graphical plot of calculated velocities. Horizontal axis is time, vertical is particle speed. Each large division represents a different set of temperature/pressure parameters.



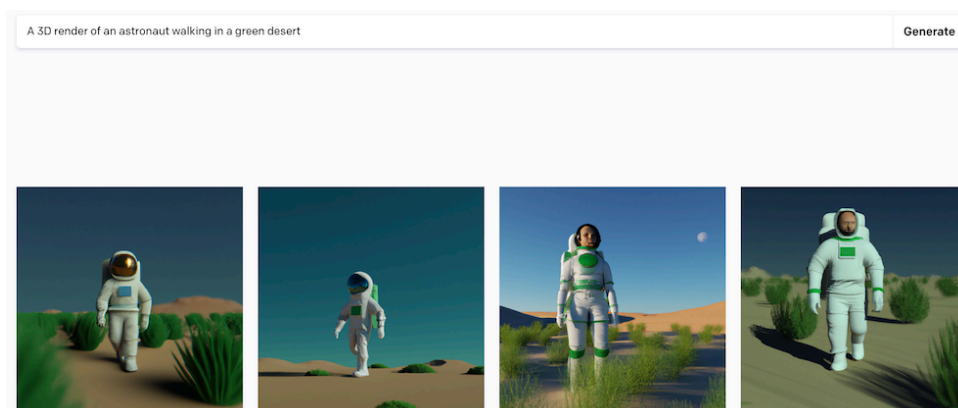
Εικόνα 4: Μαθηματική αναπαράσταση του Pithoprakta, ενός πρώιμου έργου του Ξενάκη. Μοντελοποιεί τους διαφορετικούς ηχητικούς χώρους μέσω αυτών των Glissandi (το γλίστρημα από το ένα τονικό ύψος στο άλλο) ως γραφήματα ευθειών γραμμών (ο χρόνος στον οριζόντιο άξονα, το τονικό ύψος στον κάθετο), όπου διαφορετικές κλίσεις αντιστοιχούν σε διαφορετικούς "ηχητικούς χώρους".

Γενικότερα οι δεκαετίες του 60 και 70 ήταν περιορισμένες ως προς τον τρόπο που οι υπολογιστές δημιουργούσαν τέχνη διότι βασίζονταν σε απλούς αλγορίθμους και κανόνες. Η εστίαση μετακινήθηκε από την απλούστερη, αλγοριθμική παραγωγή στη generative modeling. Ο Otto Laske, ένας εξέχων ερευνητής στον τομέα της ηχολογίας, περιγράφει αυτή την αλλαγή ως τη διαφορά μεταξύ ενός μουσικού ρομπότ και μιας μουσικής νοημοσύνης όπου η τελευταία αντικαθιστά το brute force της πρώτης με ένα knowledge-based σύστημα που διαθέτει την δική του επίγνωση του πως λειτουργούν τα διάφορα μουσικά στοιχεία. Αυτό που έφερε την αλλαγή στον τομέα είναι η έλευση των **Νευρωνικών Δικτύων** και της **Μηχανικής Μάθησης**. Τις τελευταίες δεκαετίες η **Βαθιά Μάθηση** έφερε την αλλαγή στον τομέα της τέχνης όπου με την δυνατότητα της μάθησης και της προσαρμογής σε μεγάλα σύνολα δεδομένων οι πλέον State of the Art εφαρμογές χρησιμοποιούν αποκλειστικά το Deep Learning.

1.3.2 ΑΙ και Τέχνη σήμερα

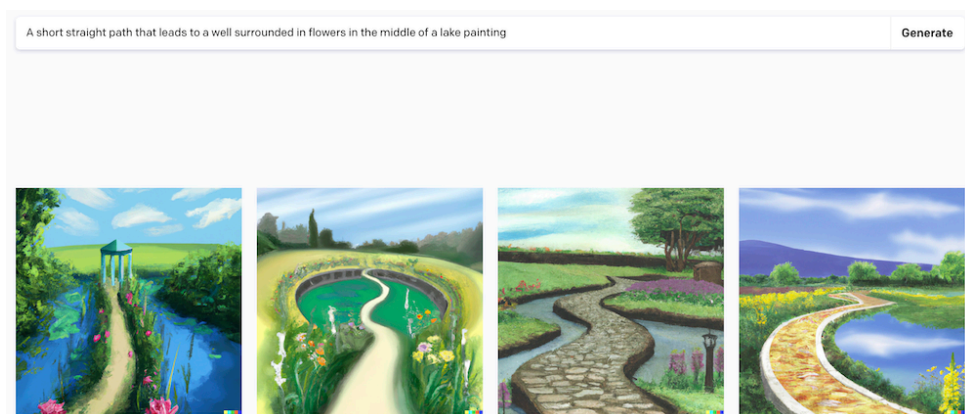
Η Τεχνητή Νοημοσύνη έχει φέρει μια επανάσταση στην τέχνη σήμερα. Στην ζωγραφική ο πρωτοποριακός αλγόριθμος CLIP (Contrastive Language-Image Pretraining) που δημιουργήθηκε από την **OpenAI**⁶ το 2020 [9] έφερε γιγάντια εξέλιξη στον τομέα της ΤΝ και δημιούργησε τον δρόμο που ακολούθησε το ΑΙ Art. Κατά τη διάρκεια της εκπαίδευσης, το CLIP μαθαίνει να συσχετίζει εικόνες και τις αντίστοιχες περιγραφές κειμένου, αντιπαραβάλλοντάς τις με άλλα τυχαία επιλεγμένα ζεύγη. Το μοντέλο εκπαιδεύεται ώστε να συγκεντρώνει τα σωστά ζεύγη (θετικά παραδείγματα) και να απομακρύνει τα λανθασμένα ζεύγη (αρνητικά παραδείγματα). Αυτή η διαδικασία βοηθά το CLIP να αναπτύξει έναν κοινό χώρο αναπαράστασης όπου παρόμοιες εικόνες και κείμενα βρίσκονται κοντά μεταξύ τους.

Στην συνέχεια το 2022 έγινε το άλμα του ΑΙ Art στην Mainstream κουλτούρα με τα **Diffusion Models**. Παραδείγματα από Text to Image φαίνονται στις Εικόνες 5, 6.



Εικόνα 5: Παράδειγμα του Dall-e 2. Πάνω φαίνεται το text prompt με αυτό που ζητήθηκε να δημιουργήσει

⁶<https://openai.com/research/clip>



Εικόνα 6: Δεύτερο παράδειγμα.

Ακόμη μια γνωστή εφαρμογή στον τομέα του Image Generation είναι το **Midjourney**⁷ το οποίο κι αυτό χρησιμοποιεί τον αλγόριθμο Stable Diffusion για να δημιουργήσει εικόνες. Εκτός από την δυνατότητα να δημιουργήσει εικόνες μπορεί να συνδιάσει εικόνες. Στις Εικόνες 7, 8 φαίνεται το Midjourney σε δράση.



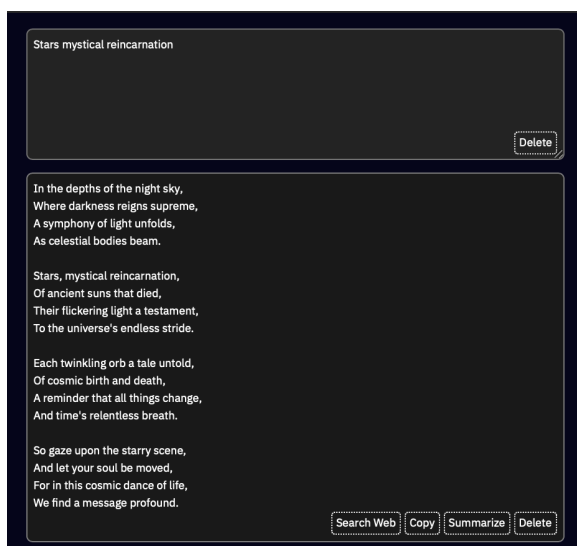
Εικόνα 7: Théâtre D'opéra Spatial, μια εικόνα του Midjourney που κέρδισε το πρώτο βραβείο σε διαγωνισμό ψηφιακής τέχνης.

⁷<https://www.midjourney.com>

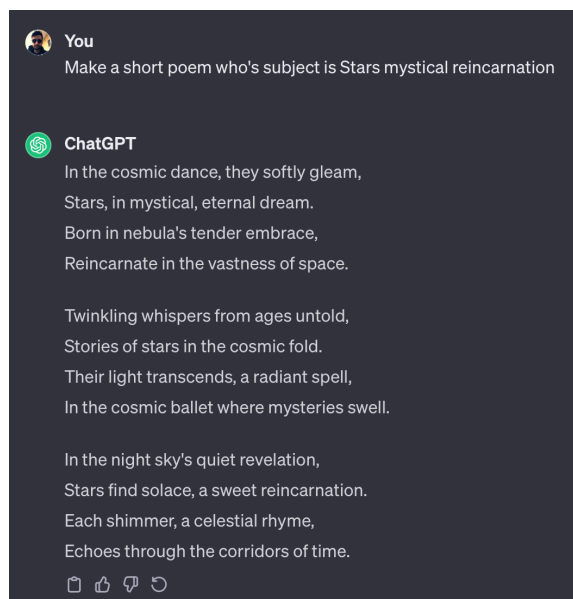


Εικόνα 8: Μια ψεύτικη εικόνα του Πάπα Φραγκίσκου που δημιούργησε το Midjourney, η οποία έγινε viral το 2023.

Άλλα παραδείγματα της χρήσης Τεχνητής Νοημοσύνης στην Τέχνη είναι στην ποίηση όπου με μοντέλα Text Generation όπως το ChatGPT⁸ ή πιο συγκεκριμένα προγράμματα όπως το DeepAI⁹ [10] μπορούν βάση ενός Text Prompt να δημιουργήσουν ποίηση όπως φαίνεται στην Εικόνα 9.



(i) DeepAI



(ii) ChatGPT

Εικόνα 9: Δημιουργία ποιήματος βάση περιγραφής

1.3.3 ΑΙ και Μουσική σήμερα

Από αυτό το σημείο και έπειτα θα ασχοληθούμε αποκλειστικά με τον τομέα της Μουσικής στην Τεχνητή Νοημοσύνη. Σήμερα οι τομείς που η Τεχνητή Νοημοσύνη έχει εφαρμογή στην Μουσική είναι πάμπολλοι [11].

⁸<https://chat.openai.com>

⁹<https://deepai.org/chat/poet>

- **Streaming Recommendations**

Εφαρμογές όπως το Spotify, Apple Music, Deezer κ.α. έχουν ως βασική και απαραίτητη πλέον λειτουργία το να προτείνουν μουσική παρόμοια με αυτή που ακούει ο χρήστης λαμβάνοντας χαρακτηριστικά όπως Tempo, Mood. Καλλιτέχνες, μουσικά είδη αναλύονται για να δημιουργούν κατάλληλα playlist για τους χρήστες και λαμβάνονται keywords μέσα από Social Media ή ιστοσελίδες που αφορούν τραγούδια ώστε να προσθέσουν επιπλέον χαρακτηριστικά σε ένα κομμάτι όπως π.χ. το επίπεδο Χαράς που έχει έτσι ώστε για έναν χρήστη που ακούει Χαρούμενα τραγούδια να του προταθεί το συγκεκριμένο κομμάτι.

- **Song Writing**

Η σύνθεση μουσικής με τεχνητή νοημοσύνη λειτουργεί με την εκπαίδευση αλγορίθμων μηχανικής μάθησης σε μεγάλα σύνολα δεδομένων υπάρχουσας μουσικής, προκειμένου να δημιουργηθούν μουσικές που ακούγονται παρόμοιες με αυτές του συνόλου δεδομένων. Ο χρήστης επιλέγει τη διάθεση, το είδος και τον ρυθμό και το μοντέλο δημιουργεί ένα "ολοκαίνουργιο" κομμάτι. Παραδείγματα γνωστών εφαρμογών είναι το Boomy¹⁰ το οποίο από την δημιουργία του το 2019 έχει δημιουργήσει πάνω από 16 εκατομμύρια κομμάτια.

Άλλη εφαρμογή είναι το Riffusion¹¹ το οποίο με ένα Text Prompt δημιουργεί μουσική και χρησιμοποιεί για στοίχους το text που του εισήγαγε ο χρήστης.

Άλλες εφαρμογές περιλαμβάνουν Loudly¹², το Amper Music και πολλά άλλα.

Στον τομέα του Song Writing εμπεριέχονται και μοντέλα όπου δεδομένου μιας εισόδου που είναι η φωνή ενός ανθρώπου μπορούν να την τροποποιήσουν και να την χρησιμοποιήσουν στην θέση της αρχικής φωνής ενός τραγουδιού. Αυτό ονομάζεται Vocal Cloning και γνωστή εφαρμογή είναι το voice-swap¹³.

- **AI Sound Changer** Ένα μοντέλο Sound Changer αναλύει μια μελωδία και την αντικαθιστά με ένα διαφορετικό όργανο. Για παράδειγμα μπορεί ο χρήστης να ανεβάσει μια ηχογράφιση από φωνή και να μετατραπεί σε βιολί ή σαξόφωνο. Το μεγάλο πλεονέκτημα του συγκεκριμένου μοντέλου είναι ότι ένας παραγωγός δεν χρειάζεται να ψάξει για κιθαρίστα ή τρομπετίστα για το κομμάτι του, μπορεί ο ίδιος να ανεβάσει μια μελωδία που την παίζει σε κάποιο όργανο και να το αντικαταστήσει με το όργανο που επιθυμεί. Μια γνωστή εφαρμογή του παραπάνω είναι το Magenta της Google Research ¹⁴.

- **AI Mastering** Ίσως η πιο χρονοβόρα εργασία στον τομέα του Sound Engineering είναι

¹⁰<https://boomy.com>

¹¹<https://www.riffusion.com>

¹²<https://www.loudly.com/ai-music-generator>

¹³<https://www.voice-swap.ai>

¹⁴<https://sites.research.google/tonetransfer>

το Mastering ενός κομματιού το οποίο μπορεί να αποτελείται από δεκάδες όργανα όπου το καθένα πρέπει να τροποποιηθεί ώστε να βρίσκεται σε συμφωνία με την υπόλοιπη μπάντα. Τα μοντέλα Mastering μπορούν να αναλύουν αρχεία ήχου και να ρυθμίζουν αυτόματα τα επίπεδα, το EQ και άλλες παραμέτρους για τη δημιουργία ενός ισορροπημένου ήχου.

Γνωστές εφαρμογές πάνω στον τομέα είναι Landr¹⁵, Masterchannel¹⁶ κι άλλες.

- **AI Music Separation** Ένας τομέας της μουσικής όπου ο χρήστης ανεβάζει ένα μουσικό κομμάτι και το μοντέλο του διαχωρίζει το κάθε όργανο (κιθάρα, φωνή, μπάσο, κρουστά) από το κομμάτι. Ένας μπασίστας που θέλει να εκτελέσει ένα τραγούδι το οποίο έχει ήδη μπάσο μέσα αρκεί να το ανεβάσει στο μοντέλο, το μοντέλο θα του απομονώσει την μουσική χωρίς το μπάσο και ο χρήστης θα πατήσει πάνω του για να δημιουργήσει την δική του εκδοχή του τραγουδιού.

Γνωστές εφαρμογές είναι: Lalal.ai¹⁷, Vocalremover¹⁸ κ.α..

- **Lyrics Generators** Η δημιουργία στίχων βάση συναισθήματος, λέξεων κλειδιών, μουσικής και πολλών άλλων.

Εφαρμογές: Aiseo, Toolbaz.

Στο EuroPython Conference Ιούλιος 2023¹⁹ έγινε μια παρουσίαση με τις νέες εξελίξεις της Τεχνητής Νοημοσύνης στον τομέα της Μουσικής όπου μερικές εφαρμογές παρουσιάζονται παρακάτω.

- **PedalBoard Simulator** Ένα Pedalboard όπως φαίνεται και στην Εικόνα 10 είναι αυτόνομα κυκλώματα που σκοπός τους είναι να αλλάζουν τον ήχο ενός οργάνου π.χ. μιας ηλεκτρικής κιθάρας, το σήμα της κιθάρας εισέρχεται σαν είσοδος στο πετάλι και εξέρχεται με διαφορετικό ήχο.

Η εφαρμογή Pedalboard²⁰ η οποία είναι γραμμένη σε Python και είναι OpenSource κάνει ακριβώς αυτό που περιγράψαμε παραπάνω αλλά χωρίς την ανάγκη να αγοράσει ο μουσικός πετάλι, μπορεί μέσω εντολών που φαίνονται στο Κώδικα 1 να δημιουργήσει το δικό του custom πετάλι και εφέ που επιθυμεί για το όργανο του.

¹⁵<https://www.landr.com>

¹⁶<https://masterchannel.ai>

¹⁷<https://www.lalal.ai>

¹⁸<https://vocalremover.org/el/>

¹⁹<https://ep2023.europython.eu/session/music-information-retrieval-with-python>

²⁰<https://github.com/spotify/pedalboard>



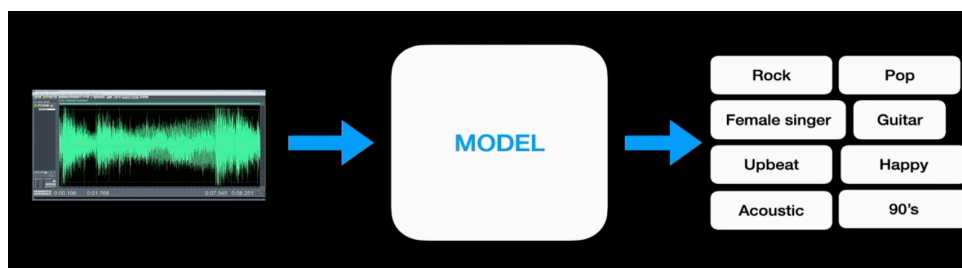
Εικόνα 10: Pedalboard Κιθάρας

```

1 from pedalboard import Pedalboard, Reverb, Compressor
2
3 board = PedalBoard([
4     Compressor(threshold_db = -50, ratio = 25),
5     Reverb(room_size=0.25),
6 ])
7
8 effected = board(audio, sample_rate)
    
```

Κώδικας 1: Κώδικας δημιουργίας του Pedalboard, το συγκεκριμένο περιέχει Εφέ Compressor και Reverb

- **Music Tagging** Όπως θα δούμε και παρακάτω στο Genre Classification το Music Tagging είναι να εισάγουμε σαν είσοδο ένα κύμα όπως φαίνεται στην Εικόνα και να μας παράξει μια ή παραπάνω ετικέτες ή αλλιώς **tag** οι οποίες χαρακτηρίζουν το τραγούδι με βάση το Είδος του, την Χρονολογία του, αν έχει άντρα ή γυναίκα τραγουδιστή όπως φαίνονται και στην Εικόνα 11.



Εικόνα 11: Music Tagging παράδειγμα

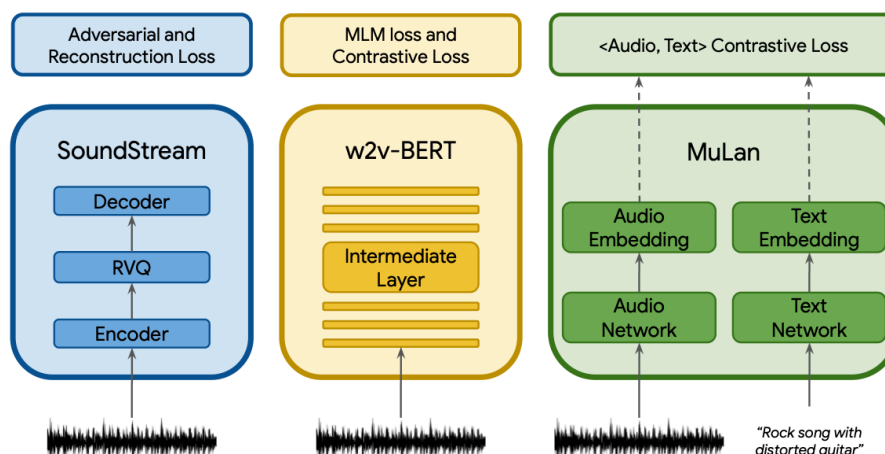
Τα State of the Art μοντέλα που επιτελούν αυτόν τον σκοπό είναι υλοποιημένα σε PyTorch που χρησιμοποιούν Βαθιά Μάθηση ²¹.

- **Text to Music Generation** Αυτά είναι μοντέλα παρόμοια του ChatGPT αλλά για μουσική τα οποία δέχονται μια περιγραφή σε μορφή κειμένου και εξάγουν την ανάλογη μουσική που ταιριάζει με την περιγραφή. Είναι αρκετά μοντέρνα και περίπλοκα μοντέλα και ένα παράδειγμα είναι το MusicLM της Google²² που βγήκε Ιανουάριο του 2023 μέσω

²¹<https://github.com/minzwon/sota-music-tagging-models>

²²<https://google-research.github.io/seanet/musiclm/examples/>

του Paper [12]. Αποτελείται από 3 μέρη Νευρωνικών Δικτύων όπως φαίνεται στην Εικόνα12, το πρώτο μέρος (SoundStream) δημιουργεί το Audio, το δεύτερο μέρος (w2v-BERT) δημιουργεί semantic tokens το οποίο σημαίνει ότι προσπαθεί να βγάλει νόημα από μια ακολουθία audio και τέλος το τρίτο μέρος (MuLan) το οποίο εξασφαλίζει ότι αυτό που δημιουργείται και το Audio που επεξεργάζεται ταιριάζει με την περιγραφή που μπήκε σαν είσοδος.



Εικόνα 12: Το μοντέλο MusicLM

Άλλη εφαρμογή πολύ πρόσφατη με την ημερομηνία που γράφεται η Διπλωματική που έχει τον ίδιο σκοπό με την προηγούμενη δημιουργήθηκε από την **Meta AI** και ονομάζεται είναι το **MusicGen**²³ και δημοσιεύτηκε 7 Νοεμβρίου το 2023 μέσω της έρευνας [13].

Τέλος, από τον άλλο σημερινό τιτάνα που μας έφερε το Dall-E και το ChatGPT, η OpenAI δημιούργησε το **Jukebox**²⁴ το οποίο το παρουσίασε με το Paper της [14] τον Απρίλιο το 2020. Το Jukebox όπως τα προηγούμενα είναι ένα Νευρωνικό Δίκτυο το οποίο παράγει μουσική σε μορφή raw audio δίνοντας του μια είσοδο όπως Είδος μουσικής, Καλλιτέχνης και Στοιχός. Το μοντέλο εκπαιδεύτηκε πάνω σε ένα σύνολο 1.2 εκατομμύριων τραγουδιών σε συνδιασμό με τους στίχους που περιείχαν.

Τα παραπάνω είναι μερικά παραδείγματα της κατάστασης που επικρατεί σήμερα στον τομέα της Μουσικής, βλέπουμε ότι οι περισσότερες εφαρμογές και εξελίξεις είναι αρκετά πρόσφατες και γίνεται φανερό ότι βρισκόμαστε στο σημείο όπου γεννιέται η μετασχηματική εποχή για τον τομέα της Μουσικής. Οι πρόσφατες εξελίξεις έχουν αποκαλύψει νέες διαστάσεις για την κατανόηση της μουσικής αλλά και έχουν ανοίξει νέες πόρτες για εφαρμογές που κάποτε θεωρούνταν αδιανόητες. Βρισκόμαστε στο κατώφλι αυτού του ανερχόμενου επιστημονικού πεδίου και οι δυνατότητες μόλις ξεπροβάλλουν και υπόσχονται πολλά περισσότερα επιτεύγματα που μετά από κάποιο σημείο θα μας φαίνονται δεδομένα αφού ο καθένας θα μπορεί να δημιουργεί μουσική όπως μπορεί να πλοηγείται στο διαδίκτυο.

²³<https://huggingface.co/spaces/facebook/MusicGen>

²⁴<https://jukebox.openai.com>

Καλύψαμε την ιστορική αναδρομή και την σήμερα κατάσταση του Music & AI και τώρα θα μπούμε στο πρακτικό κομμάτι που αφορά τις τεχνολογίες, την επιστήμη, και τους τομείς της Επιστήμης των Υπολογιστών που μας δίνουν τα εργαλεία ώστε αυτό που παρουσιάσαμε να είναι δυνατόν.

1.4 Music Information Retrieval

Όπως υπάρχει το Information Retrieval (**IR**) το οποίο περιέχει τους αλγορίθμους για να εξαχθεί βέλτιστα μια πληροφορία από μια πηγή έτσι υπάρχει και το Music Information Retrieval (**MIR**) το οποίο κάνει το ανάλογο για την μουσική. Το MIR ασχολείται πρωτίστως με την εξαγωγή χαρακτηριστικών που έχουν νόημα από ένα κομμάτι όπως θα δούμε στο κεφάλαιο 4, την δεικτοδότηση των χαρακτηριστικών και την υλοποίηση μεθόδων αναζήτησης και ανάκτησης μουσικών κομματιών [15]. Οι τεχνικές MIR έχουν αναπτυχθεί για την επίλυση προβλημάτων όπως η ταξινόμηση ειδών [16], [17], η αναγνώριση καλλιτεχνών [18] και η αναγνώριση οργάνων [19]. Από το 2005, πραγματοποιείται μια ετήσια εκδήλωση αξιολόγησης με την ονομασία Music Information Retrieval Evaluation eXchange (MIREX1) για να διευκολυνθεί η ανάπτυξη αλγορίθμων MIR.

Το MIR συνδυάζει τεχνικές από την επιστήμη των υπολογιστών, την επεξεργασία σήματος, τη μηχανική μάθηση και τη θεωρία της μουσικής για τη δημιουργία αλγορίθμων και συστημάτων ικανών να επεξεργάζονται και να κατανοούν αυτόματα το μουσικό περιεχόμενο. Σήμερα, το MIR είναι ένας δυναμικός και εξελισσόμενος τομέας, ο οποίος διαδραματίζει καθοριστικό ρόλο σε εφαρμογές όπως οι υπηρεσίες music streaming, τα συστήματα συστάσεων περιεχομένου και η ανάπτυξη καινοτόμων εργαλείων για μουσικούς και μουσικούς ερευνητές. Οι ερευνητές και οι επαγγελματίες του MIR στοχεύουν να βελτιώσουν τον τρόπο με τον οποίο αλληλεπιδρούμε και ανακαλύπτουμε τη μουσική στην ψηφιακή εποχή μέσω της εφαρμογής προηγμένων υπολογιστικών τεχνικών.

2 Βιβλιογραφική Ανασκόπηση

2.1 Music Genre Classification

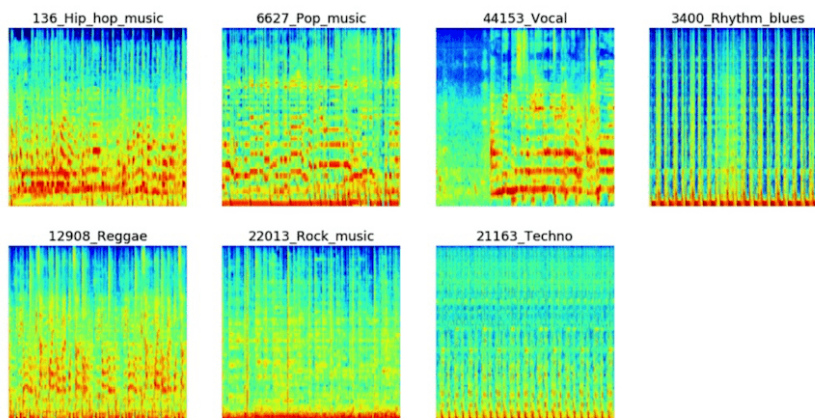
Ιστορικά μια από τις εναρκτήριες και πιο σημαντικές μελέτες στον τομέα του Music Genre Classification είναι η έρευνα του Tzanetakis *et al.* [17] το 2002 όπου αναφέρεται ότι εκείνη την εποχή το είδος της μουσικής καταγράφονταν *manually* χωρίς την χρήση κάποιου μοντέλου τεχνητής νοημοσύνης, αυτό που η έρευνα έφερε στο προσκήνιο ήταν μια μέθοδος όπου από το σήμα ήχου της μουσικής εξάγονταν τρία feature sets που απεικόνιζαν την ηχοχρωματική υφή, το ρυθμικό περιεχόμενο και το pitch, έπειτα με βάση αυτά τα δεδομένα εκπαιδεύονταν ταξινομητές (*classifiers*) στατιστικής αναγνώρισης προτύπων. Σύμφωνα με την έρευνα «*Η εξαγωγή χαρακτηριστικών είναι η διαδικασία υπολογισμού μιας συμπαγούς αριθμητικής αναπαράστασης που μπορεί να χρησιμοποιηθεί για τον χαρακτηρισμό ενός τμήματος ήχου. Ο σχεδιασμός περιγραφικών χαρακτηριστικών για μια συγκεκριμένη εφαρμογή αποτελεί την κύρια πρόκληση στην κατασκευή συστημάτων αναγνώρισης προτύπων. Μόλις εξαχθούν τα χαρακτηριστικά μπορούν να χρησιμοποιηθούν τυπικές τεχνικές μηχανικής μάθησης που είναι ανεξάρτητες από τη συγκεκριμένη περιοχή εφαρμογής.*». Η συγκεκριμένη προσέγγιση παραδοσιακής μηχανικής μάθησης που χρησιμοποιούσε ένα μοντέλο Gaussian Mixture και k-NN ταξινομητή έφτανε σε ένα Accuracy **61%** όταν από ένα ηχητικό σήμα καλούταν να ταξινομήσει το κομμάτι σε ένα από 10 Μουσικά Είδη.

Το παραπάνω είναι ένα αντιπροσωπευτικό παράδειγμα όπου λαμβάνοντας ένα ηχητικό σήμα οι ερευνητές προσπαθούσαν να βρουν ποια είναι τα πιο χρήσιμα και αντιπροσωπευτικά feature έτσι ώστε να τα χρησιμοποιήσουν σαν δεδομένα εκπαίδευσης σε μοντέλα μηχανικής μάθησης. Σε μια άλλη έρευνα [20] όπου δίνεται πάλι έμφαση σε παραδοσιακές μεθόδους ML η ακρίβεια στο MGC ήταν **k-NN 69.7%**, **GMM 64.3%** και **SVM 61.2%**. Τα αποτελέσματα αυτά δείχνουν ότι, σε γενικές γραμμές, η μέθοδος ταξινόμησης με τη χρήση ML πρέπει ακόμη να βελτιωθεί. Ένα αρνητικό που έχουν οι ML μέθοδοι είναι ότι σε πολύ μεγάλα δεδομένα αρχίζει η απόδοση τους να πέφτει κατά πολύ, και με την άνοδο που υπάρχει κάθε χρόνο σε μουσικά κομμάτια που ανεβαίνουν στο διαδίκτυο αυτές οι μέθοδοι θα αρχίσουν σιγά σιγά να θεωρούνται παρωχημένες.

Πλέον η προσέγγιση έχει αλλάξει και έχει στραφεί στην αποκλειστική χρήση των Νευρωνικών Δικτύων. Δύο βασικοί λόγοι που γίνεται αυτό είναι γιατί τα Νευρωνικά Δίκτυα έχουν το λεγόμενο **end-to-end**, το end-to-end σημαίνει ότι το Δίκτυο εκπαιδεύεται για περίπλοκα tasks παρουσιάζοντας του μόνο το raw input data και το τι θέλουμε το Δίκτυο να εξάγει. Έτσι δεν χρειάζεται *manually* να βρίσκουμε τα features που χαρακτηρίζουν ένα κομμάτι για να εκπαιδεύσουμε το μοντέλο αλλά το μοντέλο το κάνει μόνο του. Επίσης τα NN εκπαιδεύονται πάνω σε ένα πολύ μεγάλο σύνολο δεδομένων, εκεί όπου οι ML προσεγγίσεις αρχίζουν να αποτυγχάνουν.

Στην έρευνα [1] γίνεται μια σύγκριση μεταξύ παραδοσιακού Machine Learning και Νευρωνικών Δικτύων για το Music Genre Classification. Το Data Set που χρησιμοποιεί είναι μουσικά κομμάτια των 10 δευτερολέπτων από 2.1 Εκατομμύρια τραγούδια του Youtube. Αναφέρει πως για την προσέγγιση του Deep Learning δεν χρειάζονται «hand-made» χαρακτηριστικά.

Για νευρωνικό δίκτυο χρησιμοποιούν **Convolutional Neural Network** όπου σαν δεδομένα εισόδου εισάγονται τα spectrograms όπως φαίνονται στην εικόνα 13.



Εικόνα 13: Spectrograms για τα Music Genre της έρευνας [1]

Στην παραδοσιακή Machine Learning προσέγγιση χρησιμοποιήθηκαν οι Classifiers **Logistic Regression, Random Forest, Gradient Boosting** και **Support Vector Machines** εκπαιδευμένα στα παρακάτω χαρακτηριστικά:

1. Time Domain Features

- Central Moments
- Zero Crossing Rate (ZCR)
- Root Mean Square Energy (RMSE)
- Tempo

2. Frequency Domain Features

- Mel-Frequency Cepstral Coefficients (MFCC)
- Chroma Features
- Spectral Centroid
- Spectral Band-width
- Spectral Contrast
- Spectral Roll-off

Τα αποτελέσματα της παραπάνω έρευνας είναι ότι με χρήση CNN υπήρχε Accuracy **64%**²⁵, και με το καλύτερο των παραδοσιακών μεθόδων ML που ήταν το Extreme Gradient Boosting είχε **59%**. Από την έρευνα μπορούμε να συμπεράνουμε ότι τα Νευρωνικά Δίκτυα έχουν παραπάνω επίδοση έναντι των ML προσεγγίσεων. Μια ακόμη χρήσιμη πληροφορία που μας προσφέρει η έρευνα είναι το ποια εν τέλη ήταν τα πιο σημαντικά χαρακτηριστικά. Τα πειράματά έδειξαν ότι οι MFCC συμβάλλουν σημαντικά σε αυτό το έργο της ταξινόμησης μουσικών

²⁵Στην παρούσα εργασία με χρήση CNN πετυχαίνουμε Accuracy 72%

ειδών. Ο μέσος όρος και η τυπική απόκλιση των spectral contrasts στις διάφορες ζώνες συχνότητας είναι επίσης σημαντικά χαρακτηριστικά. Το tempo επίσης εμφανίζεται επίσης στα 20 καλύτερα χρήσιμα χαρακτηριστικά στην θέση 12.

Έχει διεξαχθεί σημαντική έρευνα σχετικά με την εξαγωγή spectral χαρακτηριστικών, τα οποία περιέχουν κρίσιμες πληροφορίες για την ταξινόμηση ειδών, από μουσικά κλιπ και την τροφοδότηση αυτών των χαρακτηριστικών σε μοντέλα εκπαίδευσης. Ειδικότερα, πρόσφατες μελέτες υποστηρίζουν ότι η ακρίβεια ταξινόμησης μπορεί να βελτιωθεί με την ταυτόχρονη χρήση πολλαπλών spectral χαρακτηριστικών. Στην έρευνα [2] γίνεται μια σύγκριση των διάφορων Spectral Features χρησιμοποιώντας CNN. Η έρευνα αναφέρει πως τα πιο δημοφιλή χαρακτηριστικά που εξάγονται από ένα μουσικό κομμάτι είναι τα:

- Short-term Fourier transform (STFT)
- Mel-frequency cepstral coefficient (MFCC)
- Mel-spectrogram (MLS)
- Tempo
- Chromagram

Στις αρχικές προσεγγίσεις πάνω στο θέμα του Music Genre Classification (MGC) χρησιμοποιούσαν ένα από τα παραπάνω χαρακτηριστικά αλλά οι πιο πρόσφατες έρευνες προτείνουν να γίνεται ένας συνδυασμός των παραπάνω χαρακτηριστικών [21], [22], [23]. Εκεί πρώτα πρέπει να γίνεται η επιλογή των spectral χαρακτηριστικών και του τρόπου συνδυασμού τους αφού μπορεί να βελτιώσει σημαντικά την ακρίβεια του MGC και δεύτερον, η αρχιτεκτονική του μοντέλου θα πρέπει να είναι σε θέση να χειρίζεται ταυτόχρονα πολλαπλά spectral χαρακτηριστικά. Στην έρευνα [2] προτείνεται μια μεθοδολογία για να συνδυασθούν 3 spectral χαρακτηριστικά ώστε να εισαχθούν σε ένα CNN μοντέλο. Οι δύο βασικές στρατηγικές με βάση την έρευνα είναι οι **Early Fusion** και **Late Fusion**.

1. **Early Fusion.** Τα spectral χαρακτηριστικά ενσωματώνονται σε μια ενοποιημένη πηγή δεδομένων. Παρά την απλοϊκότητα της προσέγγισης σε αυτή την μέθοδο είναι δύσκολη η εξαγωγή μοναδικών χαρακτηριστικών του κάθε spectral feature. Συγκεκριμένα, όταν χρησιμοποιούνται STFT, MLS και MFCC, το early fusion μπορεί να παράγει περιττές πληροφορίες, δεδομένου ότι τα spectral χαρακτηριστικά με MLS προέρχονται από STFT και εκείνα με MFCC προέρχονται από MLS. Κατά συνέπεια, τα οφέλη από τη χρήση πολλαπλών φασματικών χαρακτηριστικών μπορεί να εξαφανιστούν λόγω του υπερβολικού πλεονασμού, ο οποίος θα μπορούσε ενδεχομένως να παραπλανήσει το μοντέλο.
2. **Late Fusion.** Εδώ τα δεδομένα ενοποιούνται ένα βήμα πριν τη φάση του classification. Συγκεκριμένα, τα δεδομένα εισόδου με πολυτροπικά χαρακτηριστικά τροφοδοτούνται στο μοντέλο CNN ξεχωριστά και η έξοδος από κάθε τρόπο συνδυάζεται. Ειδικότερα, η μέθοδος αυτή είναι αποτελεσματική στην εξαγωγή των μοναδικών χαρακτηριστικών κάθε τρόπου. Επιπλέον, όταν οι πηγές δεδομένων διαφέρουν ως προς τους ρυθμούς

δειγματοληψίας, τις διαστάσεις των δεδομένων και τις μονάδες μέτρησης, είναι μια πιο απλή και ευέλικτη προσέγγιση από την πρώιμη μέθοδο σύντηξης.

Στην Εικόνα 15 φαίνεται η αρχιτεκτονική του Late Fusion μοντέλου που αναφέραμε παραπάνω. Στην έρευνα αναφέρονται διάφορα είδη Νευρωνικών Δικτύων όπως Graph Neural Network, Attention Model αλλά αυτό που έχει την καλύτερη απόδοση σε σχέση με τα άλλα είναι το CNN. Στην Εικόνα 14 φαίνεται ένας πίνακας με τα αποτελέσματα πάνω σε 12 DataSet χρησιμοποιώντας διάφορους συνδυασμούς από STFT, MLS και MFCC. Βλέπουμε ότι η καλύτερη απόδοση υπάρχει όταν και τα τρία μαζί συνδυάζονται με το ανώτατο Accuracy να είναι **85,49%**. Είναι επίσης σημαντικό να σημειωθεί ότι η απόδοση του μοντέλου είναι η χαμηλότερη όταν χρησιμοποιείται μόνο ένα από τα spectral χαρακτηριστικά σε όλα τα σύνολα δεδομένων. Ωστόσο, η αύξηση του αριθμού των χρησιμοποιούμενων spectral χαρακτηριστικών οδηγεί σε αντίστοιχη βελτίωση της απόδοσης του βασικού μοντέλου.

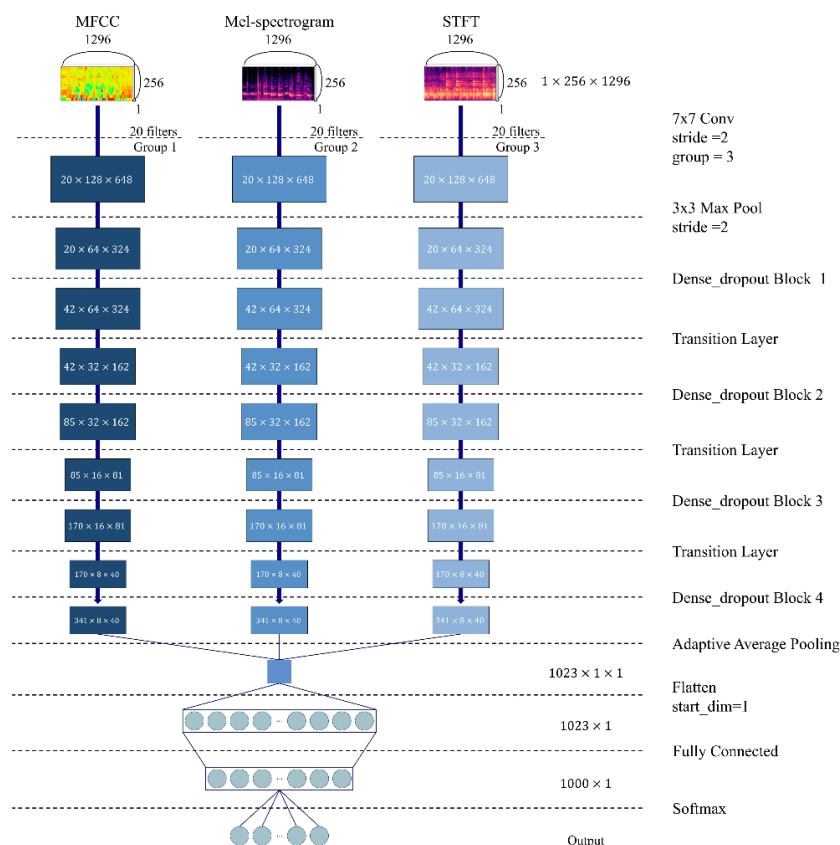
Dataset	STFS, MLS, MFCC	STFT	MLS	MFCC	STFT, MLS	MLS, MFCC	MFCC, STFT
Ballroom	58.29% ± 1.91	56.43% ± 4.12	53.36% ± 2.10	46.71% ± 3.90	57.36% ± 3.63	57.71% ± 3.12	57.71% ± 3.08
Ballroom-Extended	85.49% ± 1.06	83.25% ± 1.16	82.70% ± 1.19	78.29% ± 1.97	83.55% ± 0.99	85.48% ± 0.67	83.84% ± 1.89
EmoMusic-G	36.91% ± 0.95	34.03% ± 2.22	31.21% ± 3.04	33.83% ± 2.83	33.29% ± 2.01	34.30% ± 3.43	33.83% ± 2.31
Emotify-G	74.88% ± 2.53	73.63% ± 3.03	70.88% ± 2.57	68.13% ± 2.96	71.88% ± 4.80	73.13% ± 2.72	70.63% ± 3.74
FMA-SMALL	57.33% ± 0.94	56.00% ± 1.01	54.78% ± 0.90	45.81% ± 0.98	45.55% ± 0.88	45.11% ± 1.09	45.51% ± 0.63
GiantStepsKey-G	37.27% ± 1.80	35.12% ± 2.92	32.15% ± 2.87	29.50% ± 3.57	35.21% ± 3.60	33.72% ± 3.32	35.29% ± 2.53
GMD-G	67.13% ± 1.13	59.90% ± 2.24	64.11% ± 2.90	60.67% ± 2.37	63.78% ± 1.89	64.11% ± 2.24	62.25% ± 2.63
GTZAN	82.00% ± 2.24	74.75% ± 2.19	80.10% ± 1.26	60.20% ± 1.67	82.00% ± 1.73	79.15% ± 2.65	78.45% ± 2.70
HOMBURG	58.76% ± 1.15	57.22% ± 1.16	56.64% ± 2.31	46.72% ± 1.72	58.47% ± 1.47	55.45% ± 1.93	56.48% ± 1.90
ISMIR04	80.59% ± 1.41	75.07% ± 2.23	75.09% ± 1.43	74.63% ± 1.54	77.69% ± 1.00	78.97% ± 1.89	79.41% ± 2.23
MICM	43.25% ± 2.32	41.18% ± 2.21	40.22% ± 2.61	40.88% ± 3.19	41.36% ± 1.83	39.65% ± 2.85	41.32% ± 2.03
Seyerlehner:Unique	72.57% ± 1.19	71.36% ± 1.70	71.56% ± 1.10	66.57% ± 1.09	72.33% ± 1.14	72.34% ± 1.56	72.41% ± 1.37
Avg. Rank	1.00	4.42	4.83	6.25	3.58	3.83	3.83

Εικόνα 14: Αποτελέσματα της έρευνας [2]

Μια πρόσφατη έρευνα του 2024 [24] προτείνει την εφαρμογή **Bidirectional Long Short-Term Memory** Νευρωνικού Δικτύου με χρήση MFCC. Το Data Set ήταν το GTZAN και πραγματοποιήθηκαν preprocessing πάνω στα κομμάτια όπου αφαιρέθηκαν τα σιωπηλά μέρη και επιμηκύνθηκαν άλλα έτσι ώστε να παραχθεί κανονικοποιημένη έξοδος. Η έρευνα πέτυχε **93,10%** ακρίβεια.

Η μελέτη [25] εφάρμοσε το μοντέλο MCLNN σε διάφορα σύνολα δεδομένων, συμπεριλαμβανομένων του GTZAN. Στα σύνολα δεδομένων, κάθε μουσικό κομμάτι θα εξαχθεί χρησιμοποιώντας το Mel-Spectrogram και θα εκπαιδευτεί και θα δοκιμαστεί χρησιμοποιώντας το 10-fold cross-validation technique. Τα αποτελέσματα που λαμβάνονται στην παρούσα μελέτη είναι ο μέσος όρος των αποτελεσμάτων της 10πλής διασταυρούμενης επικύρωσης, όπου το σύνολο δεδομένων GTZAN παράγει ακρίβεια **85,10%**.

Μια άλλη πρόσφατη έρευνα [26] χρησιμοποιεί διάφορα μοντέλα ML και ένα μοντέλο DL τα οποία θα χρησιμοποιηθούν σε τρία σύνολα δεδομένων, συμπεριλαμβανομένων του GTZAN. Το μοντέλο μάθησης DL παίρνει τα καλύτερα αποτελέσματα μεταξύ των μοντέλων ML και το μοντέλο είναι ένα νευρωνικό δίκτυο (NN) με έναν αλγόριθμο εκπαίδευσης, τον trainingdx. Πριν από την εκπαίδευση και τη δοκιμή του μοντέλου με 10-fold cross-validation, το σύνολο δεδομένων θα εξαχθεί με τη χρήση της βελτιστοποίησης African Buffalo Optimization (ABO). Τα αποτελέσματα που προέκυψαν από τη μέση ακρίβεια σε κάθε σύνολο δεδομένων ήταν **83,33%** στο σύνολο δεδομένων GTZAN.



Εικόνα 15: Αρχιτεκτονική του Late Fusion μοντέλου από το [2]

Με βάση διάφορες μελέτες που έχουν διεξαχθεί σε αντικείμενα μουσικών ειδών, η χρήση της DL προσέγγισης δίνει καλύτερα αποτελέσματα από την ML προσέγγιση.

2.2 Recommender Systems

Ο τομέας του Recommender Systems είναι ένα πολύ βασικό συστατικό στις σημερινές εφαρμογές όπως *Spotify*, *Deezer*, *Soundcloud* κ.α.. Σήμερα όπως δείχνει η έρευνα [4] οι άνθρωποι ακούν μουσική συχνότερα από οποιαδήποτε άλλη δραστηριότητα όπως τηλεόραση, ανάγνωση βιβλίων, ταινίες. Τα Συστήματα Συστάσεων στην Μουσική βοηθούν τους χρήστες να ανακαλύπτουν μουσική ανάλογα με τις προτιμήσεις τους και να αποφεύγουν μουσική που δεν τους ενδιαφέρει. Ένα καλό Σύστημα Συστάσεων είναι ικανό να αναγνωρίζει τις προτιμήσεις του χρήστη και να δημιουργεί μια λίστα από κομμάτια που ταιριάζουν με τις προτιμήσεις του. Σήμερα υπάρχουν διάφορες προσεγγίσεις πάνω σε αυτόν τον τομέα που κυρίως περιλαμβάνουν το **Context-Based filtering**, το **Content-Based filtering**, το **Collaborative Filtering** και Υβριδικές που συνδυάζουν πάνω από ένα.

2.2.1 Context-Based

Σε αυτή την μεθοδολογία λαμβάνονται υπόψη στην σύσταση οι συγκυριακοί παράγοντες όπως ώρα της ημέρας, φυσική δραστηριότητα του χρήστη, τοποθεσία στον χάρτη [27], [28]. Χρησιμοποιεί τεχνικές εξόρυξης ιστοσελίδων/εγγράφων για να φιλτράρει σημαντικές πλη-

ροφορίες για την για να βρει ομοιότητα καλλιτεχνών, η ταξινόμηση ειδών, η ανίχνευση συναισθημάτων [29]. Ορισμένοι ερευνητές έχουν προτείνει ότι η χρήση των κοινωνικών πληροφοριών έχει ξεπεράσει το Content-Based μοντέλο [30], [31]. Ωστόσο η δημοφιλής μουσική μπορεί πάντα να λάβει περισσότερες δημόσιες γνώμες από την μουσική που δεν είναι γνωστή [32]. Τελικά, η πλούσια μουσική παίρνει περισσότερη ανατροφοδότηση, αυτό οδηγεί και πάλι σε ένα πρόβλημα popularity bias.

2.2.2 Collaborative Filtering

Εδώ τα δεδομένα που χρησιμοποιούνται για την σύσταση είναι οι προτιμήσεις άλλων χρηστών με κοινά ενδιαφέροντα. Για παράδειγμα, εάν στον χρήστη 1 και στον χρήστη 2 αρέσει το ίδιο τραγούδι, το σύστημα συστάσεων μπορεί να συστήσει στον χρήστη 2 ένα άλλο τραγούδι που αρέσει στον χρήστη 1 αλλά δεν έχει ακούσει ο χρήστης 2. Η αποτελεσματικότητα της προσέγγισης είναι αμφιλεγόμενη αφού υπάρχουν έρευνες που δηλώνουν την υψηλή αποδοτικότητα της [33], ενώ άλλες που θεωρούν πως δεν είναι καλή [34].

2.2.3 Content-Based

Η κύρια ιδέα πίσω από την Content-Based προσέγγιση είναι να εξαχθεί πληροφορία για το κομμάτι απευθείας από το σήμα ήχου το οποίο ονομάζεται **Audio Content Filtering** ή από τα μεταδεδομένα **Metadata-based Filtering**. Το Audio Content Filtering βασίζεται στην ψηφιακή αναπαράσταση του ακουστικού σήματος. Στη συνέχεια, η εξαγωγή χαρακτηριστικών εκτελείται σε κάθε frame με κάποιο τρόπο που εξαρτάται από την προσέγγιση. Τέλος, τα εξαγόμενα χαρακτηριστικά συνοψίζονται για κάθε κομμάτι, για παράδειγμα, με στατιστική μοντελοποίηση της κατανομής τους. Στην έρευνα του Casey *et al.* [35] που αφορά content-based μοντέλα αναφέρονται 3 πληροφορίες που μπορούν εξαχθούν για ένα κομμάτι.

1. **High-Level Music Content Description:** Μουσικές έννοιες όπως η μελωδία ή η αρμονία χρησιμοποιούνται εδώ για να χαρακτηρίσουν ένα κομμάτι. Στις αρχές του MIR, προτάθηκαν πολλά συστήματα αναζήτησης μέσω humming που προσπαθούσαν να εξάγουν μελωδικό περιεχόμενο έτσι ώστε ο χρήστης να μπορεί να αναζητήσει μουσική τραγουδώντας ή κάνοντας humming μέρος της μελωδίας. Τέτοια συστήματα αναπτύσσονται τώρα ως εμπορικές υπηρεσίες για παράδειγμα, *naiyo.com*. Μια έρευνα των μεθόδων αναζήτησης με τραγούδι διεξήχθη από τους Hu και Dannenberg [36].

«Η μελωδία είναι αυτή που μας επιτρέπει να διακρίνουμε το ένα έργο από το άλλο. Είναι η μελωδία που τα ανθρώπινα όντα είναι έμφυτα ικανά να αναπαράγουν τραγουδώντας, σιγοτραγουδώντας και σφυρίζοντας. Είναι η μελωδία που κάνει τη μουσική αξέχαστη: είναι πιθανό να θυμόμαστε μια μελωδία πολύ καιρό αφότου έχουμε ξεχάσει το κείμενό της.» [37]

Αν και πρόκειται για μια διαισθητική προσέγγιση, η εξαγωγή μελωδίας από πολυφωνικές ηχογραφήσεις, δηλαδή πολλαπλά όργανα που παίζουν διαφορετικές γραμμές ταυτόχρονα, είναι εξαιρετικά δύσκολο να επιτευχθεί. Επειδή υπάρχει μεγάλος αριθμός διαθέσιμων μουσικών ηχογραφήσεων που μπορούν να χρησιμοποιηθούν ως πρώτο στάδιο

εισόδου σε ένα σύστημα μουσικής περιγραφής υψηλού επιπέδου, αυτό αποτελεί κίνητρο για εργασίες σχετικά με την εξαγωγή μουσικών χαρακτηριστικών υψηλού επιπέδου από ηχητικό περιεχόμενο χαμηλού επιπέδου.

2. Low-Level Music Content Description: Αυτά περιέχουν:

- STFT
- Mel Spectrum
- Chromagram
- MFCC
- Spectral Flux
- Tempo

Στην τωρινή εργασία θα εστιάσουμε σε αυτό το κομμάτι του Content-Based low Level Recommender System. Στην παρακάτω Εικόνα 16 φαίνονται οι διαφορές μεταξύ των προσεγγίσεων Content-Based και Context-Based.

	<i>Content-based</i>	<i>Context-based</i>
Prerequisites	Music file	Users
Metadata required	No	Yes
Cold-start problem	No	Yes
Popularity bias	No	Yes
Features	Objective Direct Numeric	Subjective Noisy “Semantic”

Εικόνα 16: Πίνακας παρμένος από το [3]

Στην έρευνα [38] προτάθηκε μια μέθοδος για να δημιουργούνται Playlists με βάση τα content χαρακτηριστικά ενός τραγουδιού. Η μέθοδος ήταν πρώτα ότι το τραγούδι διασπάται σε μέρη, έπειτα από κάθε μέρος εξάγεται η Spectral απεικόνιση του που στην έρευνα χρησιμοποίησαν τα MFCC και στο τέλος συγκρίνουν τα spectral frames με αυτά των άλλων κομματιών χρησιμοποιώντας την Ευκλείδεια Απόσταση (EMD), η EMD υπολογίζει το έργο που χρειάζεται να γίνει για να μετατραπεί η μια απεικόνιση σε μια άλλη. Για την δημιουργία των Playlist κομμάτια τα οποία έχουν κοντινή απόσταση σε θέμα ομοιότητας χρησιμοποιώντας το K-means Clustering αλγόριθμο δημιουργούν ένα cluster. Με βάση τα χαρακτηριστικά του τραγουδιού προτείνονται και ανάλογα τραγούδια που ανήκουν στο ίδιο cluster που ανήκει και αυτό.

Στην έρευνα του Tao Li et al. [39] χρησιμοποιείται ένας συνδυασμός από feature όπου εμπεριέχονται: MFCC, Spectral Centroid, Spectral Rolloff, Spectral Flux και Wavelet Coefficients Histogram. Τα χαρακτηριστικά αποθηκεύονται σε ένα μονοδιάστατο διάνυσμα και υπολογίζεται το EMD μεταξύ των τραγουδιών.

Στην μελέτη [40] ακολουθήθηκε η προαναφερθέντα προσέγγιση αλλά προστέθηκαν και feature που έχουν να κάνουν με το Μπάσο.

Μια ακόμη προσέγγιση ήταν του Tzanetakis [41] όπου χρησιμοποιήθηκε ένας μεγάλος συνδιασμός από feature από χαρακτηριστικά ηχοχρώματος, όπως το MFCC, χαρακτηριστικά chroma, όπως το pitch, έως ρυθμικά χαρακτηριστικά, όπως η onset energy και το beat histogram. Στη συνέχεια, μετά την κανονικοποίηση όλων των χαρακτηριστικών, η ομοιότητα υπολογίστηκε από το EMD μεταξύ των διανυσμάτων.

Η τελευταία έρευνα που θα αναφέρουμε είναι η [42] όπου τα χαρακτηριστικά που επιλέχθηκαν αποτελούνται κυρίως από ρυθμικά χαρακτηριστικά και προστέθηκαν επίσης κάποια χαρακτηριστικά ηχοχρώματος για να βελτιωθεί η απόδοση. Για τα χαρακτηριστικά του ηχοχρώματος και του ρυθμού, η απόσταση μεταξύ των διανυσμάτων των χαρακτηριστικών υπολογίστηκε χωριστά και αθροίστηκαν μετά την κανονικοποίηση. Η βασική ιδέα αυτής της μεθόδου είναι η προσομοίωση της μουσικής ομοιότητας με τρόπο ομοιότητας ρυθμού και είναι μια από τις καλύτερες μεθόδους χωρίς επίβλεψη μέχρι στιγμής.

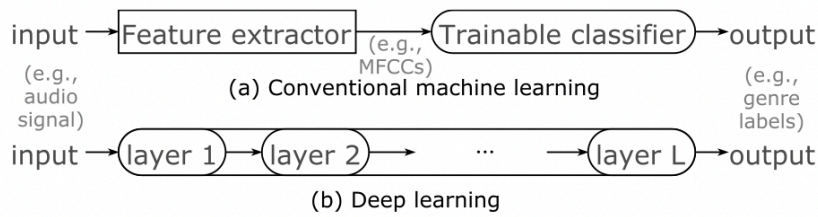
3 Deep Learning

Οι παραδοσιακές μέθοδοι μηχανικής μάθησης για την ταξινόμηση μουσικών ειδών κυρίως βασίζονται σε χαρακτηριστικά και συστήματα βασισμένα σε κανόνες τα οποία πρέπει μόνοι μας σαν μηχανικοί να εξάγουμε από κάθε Dataset στο οποίο εργαζόμαστε. Αυτή η μέθοδος απαιτεί πολύ χρόνο και μπορεί να μην καταφέρει να αποτυπώσει τους κανόνες, τα μοτίβα που ενυπάρχουν στα διάφορα μουσικά στυλ. Εισάγετε τη βαθιά μάθηση, ένα ισχυρό υποσύνολο της μηχανικής μάθησης που έχει φέρει επανάσταση σε διάφορους τομείς, συμπεριλαμβανομένου του περίπλοκου πεδίου της μουσικής ανάλυσης.

Οι τεχνικές βαθιάς μάθησης που ως το βασικότερο δομικό χαρακτηριστικό έχουν τα νευρωνικά δίκτυα, έχουν επιδείξει αξιοσημείωτη επιτυχία στην εξαγωγή σύνθετων ιεραρχικών χαρακτηριστικών από ακατέργαστα δεδομένα, καθιστώντας τις κατάλληλες για εργασίες όπως η ταξινόμηση μουσικών ειδών. Ένα από τα βασικά πλεονεκτήματα που διαθέτει η βαθιά μάθηση έναντι των παραδοσιακών προσεγγίσεων μηχανικής μάθησης είναι η ικανότητά της να μαθαίνει αυτόματα ιεραρχικές αναπαραστάσεις από τα δεδομένα, εξαλείφοντας την ανάγκη για χειροκίνητη σχεδίαση χαρακτηριστικών. Αυτή η προσαρμοστικότητα επιτρέπει στα μοντέλα βαθιάς μάθησης να διακρίνουν περίπλοκα μοτίβα και λεπτές παραλλαγές στα μουσικά είδη που μπορεί να είναι δύσκολο να καταγραφούν μόνο με τη χρήση χαρακτηριστικών που εισήχθησαν "με το χέρι".

Επιπλέον, τα μοντέλα βαθιάς μάθησης μπορούν να μαθαίνουν αυτόματα σχετικά χαρακτηριστικά στα διάφορα σύνολα δεδομένων που δέχονται σαν είσοδο, επιτρέποντάς τους να καταγράφουν τόσο λεπτομέρειες χαμηλού επιπέδου, όπως το ηχόχρωμα και ο ρυθμός, όσο και δομές υψηλού επιπέδου, όπως η εξέλιξη των συγχορδιών και η μελωδία. Αυτό όχι μόνο βελτιώνει την ακρίβεια της ταξινόμησης των ειδών, αλλά παρέχει επίσης μια πιο ολιστική κατανόηση της μουσικής που αναλύεται.

Όπως φαίνεται στην Εικόνα 17 η οποία πάρθηκε από την έρευνα [43] στο οποίο γίνεται μια σύγκριση των μεθοδολογιών παραδοσιακής Μηχανικής Μάθησης με μεθοδολογίες Βαθιάς Μάθησης. Στο Paper αναφέρεται ότι οι παραδοσιακές προσεγγίσεις στη μηχανική μάθηση συχνά εμπλέκουν τον χειροκίνητο σχεδιασμό χαρακτηριστικών και την εκπαίδευση ενός ταξινομητή από τη μηχανή, ενώ οι προσεγγίσεις βαθιάς μάθησης απαιτούν πολλαπλά εκπαιδευσιμα στρώματα που μαθαίνουν συνολικά από τα δεδομένα. Στις συμβατικές προσεγγίσεις, όπως περιγράφεται στο (α), η εκπαίδευση επικεντρώνεται στον ταξινομητή, χρησιμοποιώντας χειροκίνητα επιλεγμένα χαρακτηριστικά, όπως οι συντελεστές Mel-frequency cepstrum (MFCCs). Αντίθετα, στις προσεγγίσεις βαθιάς μάθησης, όπως φαίνεται στο (β), τα νευρωνικά δίκτυα με πολλαπλά στρώματα εξάγουν πολύπλοκες σχέσεις από τα δεδομένα, επιτρέποντας πλήρη εκπαίδευση από την είσοδο έως την έξοδο. Η πλούσια αυτή εκπαιδευτική διαδικασία χαρακτηρίζει τη βαθιά μάθηση ως μάθηση από άκρο σε άκρο, ενισχύοντας την ικανότητα ανάλυσης και κατηγοριοποίησης των ποικίλων μουσικών ειδών.

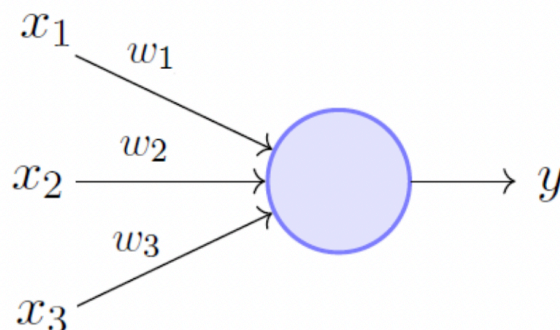


Εικόνα 17: Σύγκριση Traditional ML με DL

Σε αυτή την ενότητα, θα παρουσιάσουμε εκτενώς ό,τι χρειάζεται ο αναγνώστης για να αποκτήσει πλήρη κατανόηση του τομέα της Βαθιάς Μάθησης και των Νευρωνικών Δικτύων. Ο σκοπός είναι να παρέχουμε το θεωρητικό υπόβαθρο που απαιτείται για την κατανόηση της εφαρμογής που θα αναπτύξουμε. Αυτό το κεφάλαιο θα λειτουργήσει όχι μόνο ως εισαγωγή, αλλά και ως πηγή αναφοράς για έννοιες, αλγόριθμους και μεθόδους υλοποίησης που θα εξετάσουμε στη συνέχεια. Εντέλει, αυτή η ενότητα μπορεί να χρησιμοποιηθεί ως πηγή αναφοράς, επιτρέποντας στον αναγνώστη να επιστρέφει σε αυτόν τον τομέα σε περίπτωση που αντιμετωπίζει δυσκολίες στην κατανόηση των επόμενων κεφαλαίων.

3.1 Ο Νευρώνας

Στον τομέα της Βαθιάς Μάθησης, ο νευρώνας χρησιμεύει ως το θεμελιώδες δομικό στοιχείο, μιμούμενος τη βασική λειτουργική μονάδα του ανθρώπινου εγκεφάλου. Εμπνευσμένα από το περίπλοκο δίκτυο του βιολογικού νευρώνα, τα τεχνητά νευρωνικά δίκτυα στη Βαθιά Μάθηση στοχεύουν στην αναπαραγωγή και την ενίσχυση των γνωστικών διαδικασιών. Ο νευρώνας στο πλαίσιο της βαθιάς μάθησης είναι μια μαθηματική αφαίρεση, που λειτουργεί ως υπολογιστικός κόμβος που επεξεργάζεται και μεταδίδει πληροφορίες. Αυτοί οι τεχνητοί νευρώνες, οργανωμένοι σε επίπεδα, αποτελούν την αρχιτεκτονική των νευρωνικών δικτύων, όπου συνεργάζονται για να μαθαίνουν μοτίβα, να κάνουν προβλέψεις και να εκτελούν σύνθετες εργασίες. Η κατανόηση του ρόλου και της συμπεριφοράς των νευρώνων είναι καίριας σημασίας για την αποκάλυψη των μυστηρίων της βαθιάς μάθησης, όπου η συνέργεια των διασυνδεδεμένων τεχνητών νευρώνων διευκολύνει τις αξιοσημείωτες δυνατότητες των μοντέλων μηχανικής μάθησης.



Εικόνα 18: Ο Νευρώνας Perceptron

Η πιο βασική μονάδα στο νευρωνικό δίκτυο είναι όπως αναφέραμε ο Νευρώνας, έτσι το

πιο απλό νευρωνικό δίκτυο το οποίο αποτελείται από έναν Νευρώνα φαίνεται στην Εικόνα 18. Αυτό που φαίνεται σε μορφή Νευρώνα είναι πραγματικά μια μαθηματική πράξη που εκτελείται. Συγκεκριμένα αυτό που γίνεται είναι ότι η κάθε είσοδος x_i πολλαπλασιάζεται με το ανάλογο βάρος της w_i και όλες αυτές οι τιμές αθροίζονται έτσι ώστε να καταλήξουν σαν αποτέλεσμα στον Νευρώνα. Έτσι ο Νευρώνας (θα το συμβολίσουμε ως h το οποίο αποκαλείται επίσης Net Input) καταλήγει να έχει την τιμή:

$$h = \sum_{i=1}^n x_i \cdot w_i, \quad n : \text{Σύνολο Εισόδων } x \quad (1)$$

Αυτή η τιμή μέσα στον Νευρώνα δέχεται ακόμη μια επεξεργασία η οποία είναι ότι η τιμή h περνάει μέσα από μια συνάρτηση f η οποία δίνει την τελική τιμή y όπως φαίνεται στο σήμα η οποία είναι η έξοδος του Νευρώνα. Άρα:

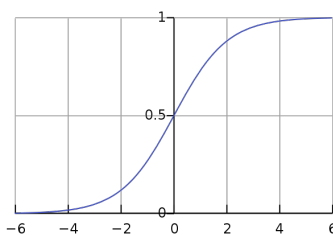
$$y = f(h) = f\left(\sum_{i=1}^n x_i \cdot w_i\right) \quad (2)$$

Η συνάρτηση f ονομάζεται **Συνάρτηση Ενεργοποίησης** και στην υλοποίηση μας θα χρησιμοποιήσουμε 3 διαφορετικές συναρτήσεις.

3.2 Συναρτήσεις Ενεργοποίησης

Οι συναρτήσεις ενεργοποίησης έχουν ως σκοπό να πάρουν την τιμή h όπως φαίνεται από τον τύπο 2 και να παράγουν μια νέα τιμή y η οποία θα διαδοθεί από τον νευρώνα στο επόμενο τμήμα του Νευρωνικού Δικτύου. Ανάλογα με την τιμή h μια συνάρτηση ενεργοποίησης λαμβάνει την απόφαση εάν ο Νευρώνας θα στείλει σήμα στον επόμενο Νευρώνα ή όχι.

1. Sigmoid Function

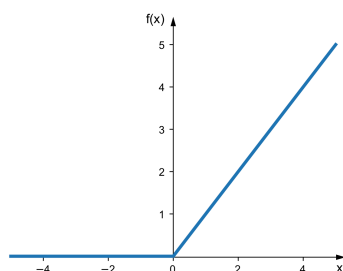


Εικόνα 19: Sigmoid Καμπύλη

Η σιγμοειδής συνάρτηση όπως φαίνεται στην Εικόνα 19 έχει τον παρακάτω τύπο 3. Χαρακτηριστικό της Sigmoid συνάρτησης είναι ότι είναι παραγωγίσιμη και παράγει μια απλή παράγωγο την οποία θα την χρησιμοποιήσει το Gradient Descent το οποίο αποτελεί αλγόριθμο εκμάθησης των Νευρωνικών Δικτύων με τα οποία θα ασχοληθούμε στην συνέχεια. Επίσης τείνει να μετριάξει έως έναν βαθμό το λεγόμενο Vanishing Gradient στο οποίο θα αναφερθούμε στο επόμενο υποκεφάλαιο.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (3)$$

2. ReLU



Εικόνα 20: ReLU Καμπύλη

Η rectified linear activation function ή εν συντομία ReLU είναι μια γραμμική συνάρτηση που εξάγεται απευθείας από την είσοδο αν είναι θετική, διαφορετικά, θα εξάγεται μηδέν. Έχει γίνει η προεπιλεγμένη συνάρτηση ενεργοποίησης για πολλούς τύπους νευρωνικών δικτύων, επειδή ένα μοντέλο που τη χρησιμοποιεί είναι ευκολότερο στην εκπαίδευση και συχνά επιτυγχάνει καλύτερες επιδόσεις. Στον τύπο 4 φαίνεται ο τρόπος που λειτουργεί η ReLU. Η ReLU ενώ φαίνεται γραμμική στην πραγματικότητα δεν είναι το οποίο είναι ένα βασικό χαρακτηριστικό που χρειάζεται να έχει μια συνάρτηση ενεργοποίησης στα Νευρωνικά Δίκτυα και το άλλο της θετικό είναι ότι εξαλείφει το πρόβλημα του Vanishing Gradient.

$$f(x) = \begin{cases} x, & \text{αν } x \geq 0 \\ 0, & \text{αν } x < 0 \end{cases} \quad (4)$$

3. SoftMax

Ο τύπος της SoftMax είναι ο 5. Η συνάρτηση SoftMax δέχεται ως είσοδο ένα διάνυσμα x από n πραγματικούς αριθμούς και το κανονικοποιεί σε μια κατανομή πιθανοτήτων που αποτελείται από n πιθανότητες ανάλογες με τα εκθετικά των αριθμών εισόδου. Δηλαδή, πριν από την εφαρμογή της SoftMax, ορισμένες συνιστώσες του διανύσματος μπορεί να είναι αρνητικές ή μεγαλύτερες της μονάδας και μπορεί να μην έχουν άθροισμα 1, αλλά μετά την εφαρμογή της SoftMax, κάθε συνιστώσα θα βρίσκεται στο διάστημα $(0,1)$ και οι συνιστώσες θα έχουν άθροισμα 1, ώστε να μπορούν να ερμηνευθούν ως πιθανότητες. Επιπλέον, οι μεγαλύτερες συνιστώσες εισόδου θα αντιστοιχούν σε μεγαλύτερες πιθανότητες.

Η SoftMax χρησιμοποιείται συνήθως στο τελευταίο στρώμα ενός Νευρωνικού Δικτύου όταν δουλεύουμε με Multi-Class Classifier δηλαδή όταν πρέπει να εξάγει ένα συμπέρασμα για πάνω από 2 κλάσεις. Αυτή θα χρησιμοποιήσουμε μιας και στην υλοποίηση μας θα πρέπει ένα τραγούδι να ταξινομηθεί σε ένα σύνολο 10 κλάσεων δηλαδή μουσικών ειδών.

$$\sigma(x_i) = \frac{e^{x_i}}{\sum_{j=1}^n e^{x_j}} \quad (5)$$

3.3 Multilayer Perceptron - Forward Propagation

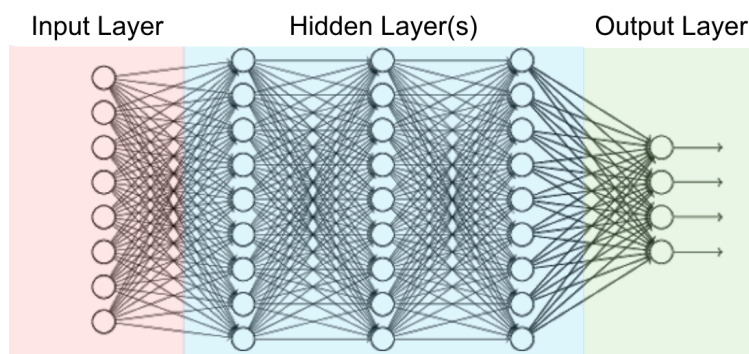
Τα multi-layer Νευρωνικά Δίκτυα είναι αυτά που χαρακτηρίζουν την Βαθιά Μάθηση, μια εικόνα ενός Deep Neural Network φαίνεται στην Εικόνα 21. Ένα Νευρωνικό Δίκτυο με πολλά

στρώματα χωρίζεται σε τρία τμήματα όπως φαίνεται στην εικόνα.

Το πρώτο τμήμα αφορά την είσοδο του Νευρωνικού Δικτύου, εκεί είναι που μπαίνουν τα δεδομένα για τα οποία θέλουμε να εξάγουμε ένα συμπέρασμα.

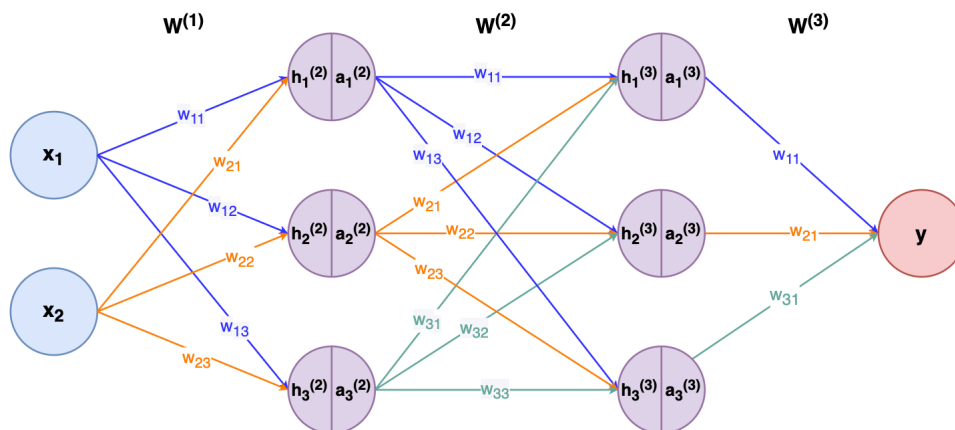
Το δεύτερο έως το προτελευταίο τμήμα του Νευρωνικού Δικτύου αποκαλείται Hidden Layer. Εκεί είναι που γίνονται όλοι οι απαραίτητοι υπολογισμοί ώστε να καταλήξουμε στο τελευταίο Layer.

Το τελευταίο Layer είναι το Output Layer το οποίο αφορά τις κλάσεις²⁶ του προβλήματος. Στην υλοποίηση της εργασίας οι 4 νευρώνες που απεικονίζονται στην εικόνα μπορούν να αφορούν 4 διαφορετικά είδη μουσικής όπου δεδομένα εισόδου εισάγονται στο Input Layer, επεξεργάζονται στα Hidden Layers και στο τέλος 1 Νευρώνας ενεργοποιείται από το Output Layer όπου θα είναι το είδος στο οποίο κατηγοριοποιήθηκε το τραγούδι της εισόδου.



Εικόνα 21: Βαθύ Νευρωνικό Δίκτυο

Θα ορίσουμε κάποιες έννοιες πάνω στο MLP οι οποίες θα είναι χρήσιμες για την συνέχεια. Στην Εικόνα 22 φαίνονται οι ονομασίες που θα χρησιμοποιούμε από εδώ και πέρα.



Εικόνα 22: MultiLayer Perceptron

Στον κώδικα κάθε επίπεδο του νευρωνικού δικτύου αποθηκεύεται στην μορφή διανυσμάτων και πινάκων. Ξεκινώντας από την είσοδο όπως φαίνεται στην Εικόνα 22 τα x_1, x_2 είναι οι νευρώνες εισόδου. Οπότε ορίζουμε ένα διάνυσμα x στην μορφή:

²⁶Στην δική μας περίπτωση του Music Genre Classification

$$x = [x_1, x_2]$$

Ή γενικότερα $x = [x_1, x_2, \dots, x_n]$, $n =$ Πλήθος νευρώνων εισόδου

Έπειτα τα βάρη του κάθε στρώματος αποθηκεύονται στον πίνακα $W^{(i)}$ όπου το superscript i δείχνει σε ποιο layer ανήκει ο συγκεκριμένος πίνακας βαρών, όποτε υπάρχει αριθμός superscript σε παρένθεση υπονοεί πάντα το layer στο οποίο ανήκει η βάση του. Στην παραπάνω εικόνα βλέπουμε για παράδειγμα ότι το $W^{(1)}$ εμπεριέχει τα παρακάτω βάρη.

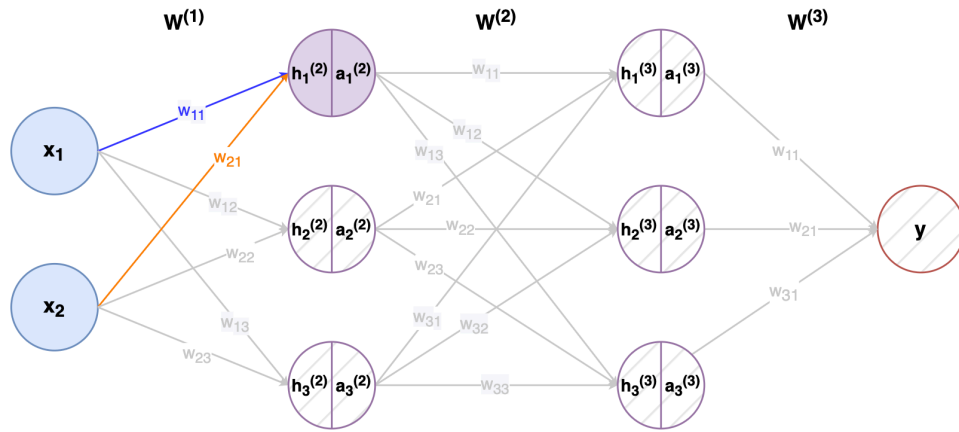
$$W^{(1)} = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{bmatrix}$$

Στην συνέχεια βλέπουμε ότι οι νευρώνες στα hidden layers αποτελούνται από δύο μέρη²⁷, αν πάρουμε το πρώτο νευρώνα αποτελείται από τα $h_1^{(2)}$ και $a_1^{(2)}$. Το $h_1^{(2)}$ προκύπτει από τον τύπο που υπολογίζεται η είσοδος ενός νευρώνα όπως είδαμε και παραπάνω στον τύπο 1. Θεωρητικά για να βρούμε το $h_1^{(2)}$ θα πρέπει να γίνει η πράξη $x_1w_{11} + x_2w_{21}$, παρομοίως για το $h_2^{(2)}$ πρέπει να γίνει η πράξη $x_1w_{12} + x_2w_{22}$. Όμως μέσω του πίνακα $W^{(1)}$ και του διανύσματος x που ορίσαμε παραπάνω μπορούμε να βρούμε όλα τα $h_i^{(2)}$ πολλαπλασιάζοντας τα:

$$\begin{aligned} h_i^{(2)} &= x \cdot W^{(1)} \\ h_i^{(2)} &= [x_1, x_2] \cdot \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{bmatrix} \\ &= [x_1w_{11} + x_2w_{21}, x_1w_{12} + x_2w_{22}, x_1w_{13} + x_2w_{23}] \\ &= [h_1^{(2)}, h_2^{(2)}, h_3^{(2)}] \end{aligned}$$

Τα $h_1^{(2)}$, $h_2^{(2)}$, $h_3^{(2)}$ είναι οι νευρώνες που φαίνονται στο πρώτο Hidden Layer, αν παρατηρήσουμε τον πολλαπλασιασμό πινάκων που έγινε παραπάνω θα δούμε ότι για κάθε νευρώνα που ανήκει σε αυτό το επίπεδο λαμβάνονται υπόψιν στον υπολογισμό της τιμής του τα βάρη και οι τιμές των νευρώνων που συνδέονται με αυτόν. Με ένα πιο ξεκάθαρο σχήμα φαίνεται στην Εικόνα 23 ποια βάρη και τιμές νευρώνων συμμετέχουν στην πράξη για να βρεθεί η τιμή του $h_1^{(2)}$.

²⁷Ο λόγος είναι καθαρά σχηματικός για διευκόλυνση της κατανόησης.



Εικόνα 23: Ενεργοποιημένες νευρώνες και βάρη για τον υπολογισμό του $h_1^{(2)}$

Στο υποκεφάλαιο 3.2 αναφερθήκαμε στις συναρτήσεις ενεργοποίησης, κάθε τιμή h_i δεν στέλνεται απευθείας στο επόμενο στρώμα νευρώνων, πρώτα περνάει από την συνάρτηση ενεργοποίησης που έχουμε ορίσει και έτσι η τιμή που παράγεται είναι η a_i , αυτή θα είναι και η τιμή που θα σταλθεί. Για αυτό τον λόγο φαίνεται στο σχήμα αριστερά πρώτα το h_i και μετά το a_i λόγω χρονικής συνοχής. Άρα:

$$a_i = f(h_i), \quad f = \{\text{Sigmoid, ReLU, SoftMax}\} \quad (6)$$

Συνεχίζοντας την εφαρμογή το τελικό διάνυσμα του δεύτερου στρώματος νευρώνων θα είναι το.

$$a^{(2)} = [a_1^{(2)}, a_2^{(2)}, a_3^{(2)}]$$

Για να βρούμε τις τιμές $h^{(3)}$ αρκεί όπως και πριν να εκτελέσουμε τον πολλαπλασιασμό:

$$\begin{aligned} h^{(3)} &= a^{(2)} \cdot W^{(2)} \\ h^{(3)} &= [a_1^{(2)}, a_2^{(2)}, a_3^{(2)}] \cdot \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix} \\ &= [a_1^{(2)}w_{11} + a_2^{(2)}w_{21} + a_3^{(2)}w_{31}, \\ &\quad a_1^{(2)}w_{12} + a_2^{(2)}w_{22} + a_3^{(2)}w_{32}, \\ &\quad a_1^{(2)}w_{13} + a_2^{(2)}w_{23} + a_3^{(2)}w_{33}] \\ &= [h_1^{(3)}, h_2^{(3)}, h_3^{(3)}] \end{aligned}$$

$$a^{(3)} = f(h^{(3)}) = f([h_1^{(3)}, h_2^{(3)}, h_3^{(3)}]) = [a_1^{(3)}, a_2^{(3)}, a_3^{(3)}]$$

Στον υπολογισμό του πρώτου hidden layer χρησιμοποιήσαμε απευθείας τις τιμές $[x_1, x_2]$ αφού αυτές σαν είσοδος δεν περνούν από Activation Function.

Για να ολοκληρώσουμε την διαδικασία βλέπουμε ότι τελευταίο layer το οποίο είναι το

Output αποτελείται από έναν νευρώνα, κι έτσι αναμένουμε να πάρουμε μόνο μια τιμή από τον πολλαπλασιασμό $a^{(3)} \cdot W^{(3)}$

$$\begin{aligned} y &= [a_1^{(3)}, a_2^{(3)}, a_3^{(3)}] \cdot \begin{bmatrix} w_{11} \\ w_{21} \\ w_{31} \end{bmatrix} \\ &= [a_1^{(3)}w_{11}, a_2^{(3)}w_{21}, a_3^{(3)}w_{31}] \\ &= h^{(4)} = y \end{aligned}$$

Κι έτσι βλέπουμε πως με πολλαπλασιασμό πινάκων φτάνουμε από την είσοδο στην έξοδο. Αυτή η διαδικασία ονομάζεται **Forward Propagation**.

Από τα παραπάνω μπορούμε να εξάγουμε δύο τύπους για τον υπολογισμό των h , a που θα μας χρησιμεύσουν.

$$h^{(i)} = a^{(i-1)} \cdot W^{(i-1)} \quad (7)$$

$$a^{(i)} = f(h^{(i)}) \quad (8)$$

Μέχρι τώρα δείξαμε τον βασικό αλγόριθμο που χρησιμοποιεί ένα Νευρωνικό Δίκτυο ώστε να βάση μιας εισόδου να παράξει μια έξοδο. Όλα αυτά όμως από μόνα τους δεν καθιστούν τα Νευρωνικά Δίκτυα μοντέλα Τεχνητής Νοημοσύνης, το βασικό χαρακτηριστικό αυτών των μοντέλων είναι η ικανότητα τους να μαθαίνουν και αυτό είναι που θα δούμε στο επόμενο υποκεφάλαιο, πως μέσω του **Back Propagation** και του **Gradient Descent** ένα Νευρωνικό Δίκτυο μπορεί να μάθει.

3.4 Back Propagation

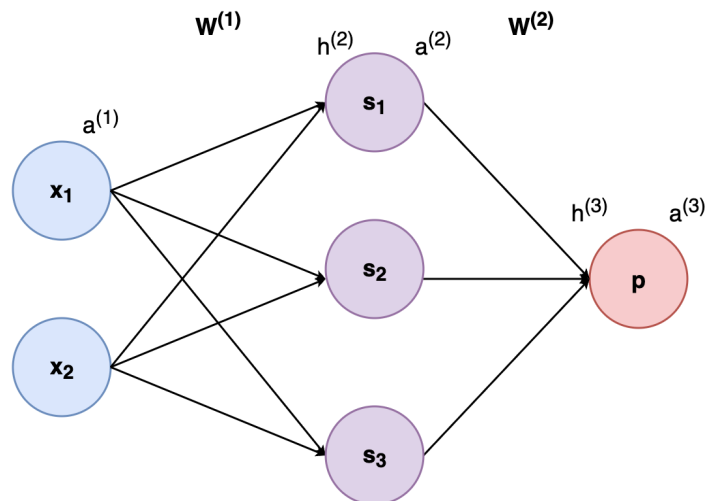
Το Back Propagation το οποίο προτάθηκε πρώτη φορά στο [44] καλείται να μετατρέψει το Νευρωνικό Δίκτυο σε ένα μοντέλο όπου οι εισόδου που του θέτουμε να μας δίνουν τα επιθυμητά αποτελέσματα, πως γίνεται αυτό; Όπως είδαμε στον τύπο 1 οι παράμετροι που λαμβάνουν μέρος στον υπολογισμό της εξόδου είναι η τιμή που έχει ο εκάστοτε νευρώνας και τα **βάρη**, αυτό που τροποποιεί το Back Propagation είναι τα βάρη του Νευρωνικού Δικτύου. Σαν μια γενική επισκόπηση αυτό που κάνει το Back Propagation είναι το εξής:

1. Το αποτέλεσμα p (*predicted*) της εξόδου του Νευρωνικού Δικτύου που βγαίνει σαν αποτέλεσμα από του Forward Propagation συγκρίνεται με την επιθυμητή έξοδο y που θα έπρεπε να έχει.
2. Βάση της διαφοράς τους αλλάζει το βάρος κάθε ακμής που υπάρχει στο δίκτυο ανάλογα με το πόσο επιρροή έχει στο τελικό αποτέλεσμα.
3. Τροποποιούνται τα βάρη, εισάγεται η επόμενη είσοδος από το training set και τα βήματα 1 και 2 επαναλαμβάνονται όσες φορές θέσει ο σχεδιαστής.

Ένας πλήρης κύκλος του συνόλου δεδομένων εκπαίδευσης που διέρχεται από ένα νευρωνικό δίκτυο, που περιλαμβάνει τόσο προς τα εμπρός όσο και προς τα πίσω περάσματα με ενημερώσεις βαρών, ονομάζεται **Epoch**. Συνήθως χρειάζονται αρκετά Epoch μέχρι το Νευρωνικό

Δίκτυο να είναι ικανοποιητικά εκπαιδευμένο.

Θα δείξουμε τώρα ένα μικρό παράδειγμα του πως λειτουργεί μαθηματικά το Back Propagation. Θα χρησιμοποιήσουμε το Δίκτυο της Εικόνας 24.



Εικόνα 24: Βαθύ Νευρωνικό Δίκτυο με 1 Hidden Layer

Αρχικά ξεκαθαρίζουμε ότι, $x_i = a_i^{(1)}$ δηλαδή οι τιμές που στέλνονται από το πρώτο layer (input) δεν παίρνουν από Activation Function αλλά είναι απευθείας οι τιμές εισόδου, και δεύτερων $a^{(3)} = p$ δηλαδή η τελική τιμή στον νευρώνα output που περνάει από activation function είναι και η τιμή p του νευρωνικού δικτύου.

Θα ορίσουμε μια **Error Function** η οποία έχει σκοπό να δείξει την διαφορά μεταξύ predicted value (p) του Δικτύου σε σχέση με επιθυμητή έξοδο (y) που επιθυμούμε.

$$E(p, y) = \frac{1}{2}(p - y)^2 \quad (9)$$

Επίσης θυμίζουμε πως βρίσκουμε τα a_i και h_i .

$$h^{(i)} = a^{(i-1)} \cdot W^{(i-1)}$$

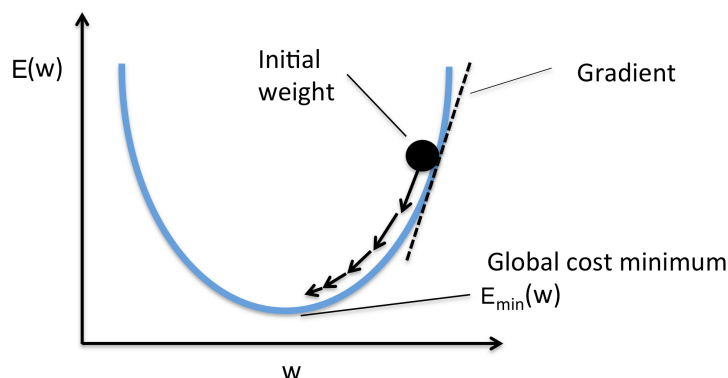
$$a^{(i)} = f(h^{(i)}), \quad f = \{\text{Sigmoid, ReLU}\}$$

Αυτό που ζητάμε είναι με βάση το Error Function E να αλλάξουμε τις τιμές των βαρών $W^{(2)}$. Για να το κάνουμε αυτό θα χρησιμοποιήσουμε μερικές παραγώγους. Με χρήση μερικών παραγώγων ψάχνουμε το 10.

$$W'^{(2)} = W^{(2)} - \eta \cdot \frac{\partial E}{\partial W^{(2)}} \quad (10)$$

Το πως προκύπτει η σχέση $\frac{\partial E}{\partial W^{(2)}}$ είναι ζήτημα του δεύτερου σημαντικού αλγορίθμου **Gradient Descent**.

3.4.1 Gradient Descent



Εικόνα 25: Gradient Descent

Το σχήμα 25 θα μας βοηθήσει πολύ στην κατανόηση αυτής της τεχνικής. Η **μπλε καμπύλη** είναι η συνάρτηση Error Function E όπου αλλάζοντας τα βάρη w βλέπουμε ότι σε ένα σημείο ελαχιστοποιείται²⁸, το οποίο σημαίνει ότι σε εκείνο το σημείο υπάρχει το μικρότερο σφάλμα στην σχέση έξοδος μοντέλου με επιθυμητή έξοδο. Εμείς στην τωρινή κατάσταση του Νευρωνικού Δικτύου μας βρισκόμαστε εκεί που είναι η **μαύρη μπάλα**²⁹ και θέλουμε να πάμε στο χαμηλό σημείο της καμπύλης, για να το κάνουμε αυτό πρέπει να βρούμε προς τα που αν αλλάξουμε το w το κόστος $E(w)$ θα μειωθεί.

Την απάντηση μας την δίνει η σχέση $\frac{\partial E}{\partial W}$ όπου επιστρέφει την φορά προς την οποία υπάρχει η **μεγαλύτερη αύξηση** του $E(w)$, ουσιαστικά αυτό που μας επιστρέφει είναι η διακεκομμένη εφαπτόμενη γραμμή που φαίνεται στο σχήμα με όνομα *gradient*, εμείς όμως επειδή θέλουμε να πάμε προς το ελάχιστο θα πάρουμε το αρνητικό της μερικής παραγώγου και έτσι προκύπτει η αφαίρεση στην εξίσωση 10. Το η που χρησιμοποιούμε για να πολλαπλασιάσουμε την μερική παράγωγο ονομάζεται **Learning Rate**³⁰ και ουσιαστικά ορίζει το πόσο μεγάλα θα είναι τα βήματα που κάνει η "μπάλα" σε κάθε επανάληψη.

Έτσι η σχέση 10 ουσιαστικά δημιουργεί το νέο πίνακα βαρών $W^{(i)}$ αφαιρώντας από τον προηγούμενο πίνακα $W^{(i)}$ την φορά προς το ελάχιστο της καμπύλης E πολλαπλασιασμένο με ένα βήμα η .

Το $\frac{\partial E}{\partial W^{(2)}}$ ανάγεται σε $\frac{\partial}{\partial W^{(2)}} \frac{1}{2}(p - y)^2$, όμως αυτό δεν μπορούμε να το υπολογίσουμε απευθείας καθώς δεν υπάρχουν όροι του παρανομαστή στον αριθμητή. Για να επιλύσουμε αυτό το πρόβλημα θα χρησιμοποιήσουμε το Chain Rule.

$$\frac{\partial E}{\partial W^{(2)}} = \frac{\partial E}{\partial a^{(3)}} \cdot \frac{\partial a^{(3)}}{\partial h^{(3)}} \cdot \frac{\partial h^{(3)}}{\partial W^{(2)}} \quad (11)$$

²⁸ Εκεί που στην εικόνα δείχνει το βελάκι, το $E_{\min}(w)$

²⁹ Στην εικόνα αναφέρεται με Initial Weight

³⁰ Αρκετοί ερευνητές θεωρούν ότι η σωστή επιλογή τιμής για το Learning Rate είναι η πιο σημαντική παράμετρος για ένα μοντέλο NN

Αν παρατηρήσουμε το NN της Εικόνας 24 τότε θα δούμε ότι ουσιαστικά εκτελούμε μια οπισθοδρόμηση από τα δεξιά προς τα αριστερά ξεκινώντας από το τελικό ποσό $a^{(3)}$, συνεχίζοντας με $h^{(3)}$ που ήταν πριν και ολοκληρώνοντας με $W^{(2)}$ που εκεί θέλουμε να καταλήξουμε. Επιλύοντας τις μερικές παραγώγους βγαίνουν τα ακόλουθα.

$$\frac{\partial E}{\partial a^{(3)}} = \frac{\partial}{\partial a^{(3)}} \left(\frac{1}{2}(a^{(3)} - y)^2 \right) = \left(\frac{1}{2}(a^{(3)} - y)^2 \right)'_{a^{(3)}} = a^{(3)} - y \quad (12)$$

$$\frac{\partial a^{(3)}}{\partial h^{(3)}} = \sigma'(h^{(3)}) = \sigma(h^{(3)})(1 - \sigma(h^{(3)})) \quad (13)$$

$$\frac{\partial h^{(3)}}{\partial W^{(2)}} = \frac{\partial}{\partial W^{(2)}} \left(a^{(2)} \cdot W^{(2)} \right) = a^{(2)} \quad (14)$$

Από τις σχέσεις 12, 13, 14 η σχέση 11 γίνεται.

$$\frac{\partial E}{\partial W^{(2)}} = \left(a^{(3)} - y \right) \cdot \sigma'(h^{(3)}) \cdot a^{(2)} \quad (15)$$

Και τα βάρη $W^{(2)}$ γίνονται:

$$W'^{(2)} = W^{(2)} - \eta \cdot \left(a^{(3)} - y \right) \cdot \sigma'(h^{(3)}) \cdot a^{(2)}$$

Αν θέλουμε να βρούμε τώρα τα βάρη $W'^{(1)}$ θα κάνουμε το ίδιο με πριν.

$$W'^{(1)} = W^{(1)} - \eta \cdot \frac{\partial E}{\partial W^{(1)}} \quad (16)$$

Και με τον αλυσιδωτό κανόνα σχηματικά από δεξιά προς τα δεξιά αναλύουμε.

$$\frac{\partial E}{\partial W^{(1)}} = \frac{\partial E}{\partial a^{(3)}} \cdot \frac{\partial a^{(3)}}{\partial h^{(3)}} \cdot \frac{\partial h^{(3)}}{\partial a^{(2)}} \cdot \frac{\partial a^{(2)}}{\partial h^{(2)}} \cdot \frac{\partial h^{(2)}}{\partial W^{(1)}} \quad (17)$$

Αν παρατηρήσουμε την σχέση 11 θα δούμε ότι οι πρώτες δύο παραγώγοι είναι κοινές με αυτές της σχέσης 17, αυτό είναι καλό γιατί βλέπουμε ότι η τροποποίηση των τιμών από τα βάρη που βρίσκονται πιο πίσω³¹ εξαρτάται από τις τιμές των βαρών που υπάρχουν μπροστά, και έτσι προκύπτει και η ονομασία *Back Propagation* γιατί σε κάθε Layer που πηγαίνουμε προς τα πίσω λαμβάνονται υπόψιν όλα τα στρώματα από τα οποία έχουμε περάσει. Οι υπόλοιπες 3 μερικές παραγώγοι βρίσκονται ως εξής.

³¹ Δηλαδή προς το layer εισόδου.

$$\frac{\partial h^{(3)}}{\partial a^{(2)}} = \frac{\partial}{\partial a^{(2)}} (a^{(2)} \cdot W^{(2)}) = W^{(2)} \quad (18)$$

$$\frac{\partial a^{(2)}}{\partial h^{(2)}} = \sigma'(h^{(2)}) \quad (19)$$

$$\frac{\partial h^{(2)}}{\partial W^{(1)}} = \frac{\partial}{\partial W^{(1)}} (a^{(1)} \cdot W^{(1)}) = a^{(1)} = x \quad (20)$$

Από τις 12, 13, 18, 19, 20 η 17 γίνεται.

$$\frac{\partial E}{\partial W^{(1)}} = (a^{(3)} - y) \cdot \sigma'(h^{(3)}) \cdot W^{(2)} \cdot \sigma'(h^{(2)}) \cdot a^{(1)} \quad (21)$$

Και έτσι καταλήγουμε να βρίσκουμε την εξίσωση που αν την υποβάλουμε στην σχέση 16 θα μας δώσει τις νέες τιμές των βαρών ώστε το δίκτυο να παράγει έξοδο πιο κοντά στην επιθυμητή τιμή. Ένα μικρό overview για το Forward-Back Propagation είναι αυτό:

1. Εισάγεται μια εγγραφή στο NN από το σύνολο δεδομένων εκπαίδευσης.
2. Με βάση τον αλγόριθμο Forward Propagation προωθούνται οι ενεργοποιήσεις μέχρι το τελευταίο layer.
3. Όταν φτάνουμε στο τελευταίο layer υπολογίσουμε το Error Function που μας δίνει την διαφορά μεταξύ έξοδο του NN και της εξόδου που θέλουμε και με βάση αυτό ξεκινάει το Back Propagation.
4. Το Back Propagation ενημερώνει όλα τα βάρη μέχρι να φτάσει στο layer εισόδου.
5. Επαναλαμβάνονται τα βήματα (1-4) με την επόμενη εγγραφή από το σύνολο δεδομένων
6. Όταν χρησιμοποιηθούν όλες οι εγγραφές από το σύνολο δεδομένων έχει ολοκληρωθεί ένα **epoch**, αν ο σχεδιαστής έχει ορίσει παραπάνω epochs τότε επανέλαβε τα βήματα (1-5) μέχρι τον προκαθορισμένο αριθμό epochs.

3.5 Είδη Gradients

Όπως είδαμε προηγουμένως ένα NN εκπαιδεύεται εκτελώντας Forward & Back Propagation πάνω σε εγγραφές από ένα σετ δεδομένων εκπαίδευσης, αυτό που χρησιμοποιείται για να αλλάξουν εν τέλη τα βάρη τιμές είναι όπως είδαμε το Gradient Descent. Στην πραγματικότητα υπάρχουν 3 διαφορετικοί τύποι Gradient Descent που χρησιμοποιούνται:

Stochastic Gradient Descent (SGD)

Σε αυτή την μέθοδο κάθε φορά εισάγεται μια εγγραφή από το σύνολο δεδομένων εκπαίδευσης και εκτελείται το Forward & Back Propagation όπου τελικά ανανεώνονται οι τιμές βαρών του NN. Το ίδιο επαναλαμβάνεται για όλες τις εγγραφές του συνόλου δεδομένων εκπαίδευσης ή

με ένα μέρος τους όπου σε αυτή την περίπτωση επιλέγονται τυχαία εγγραφές. Είναι η πιο γρήγορη μεθοδολογία από τις τρεις που θα παρουσιάσουμε αφού το μοντέλο κάθε φορά επεξεργάζεται μια εγγραφή. Ένα αρνητικό αυτής της προσέγγισης είναι ότι το SGD εισάγει θόρυβο.

Batch Gradient Descent (BGD)

Στο συγκεκριμένο όλο το σύνολο δεδομένων εκπαίδευσης εισάγεται για να υπολογιστεί το Gradient του Cost Function. Καθώς πρέπει να υπολογίσουμε το gradient σε ολόκληρο το σύνολο δεδομένων για να εκτελέσουμε μία μόνο ενημέρωση, το BGD μπορεί να είναι πολύ αργό σύνολα δεδομένων που δεν χωράνε στη μνήμη.

Mini Batch Gradient Descent (MBGD)

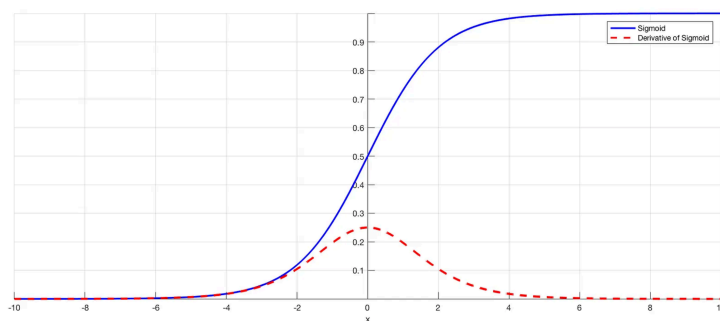
Η μέθοδος Mini-Batch Gradient Descent βρίσκει μια μέση λύση μεταξύ BGD και SGD αναλύοντας το σύνολο δεδομένων εκπαίδευσης σε μικρές τυχαίες παρτίδες (mini-batches) και υπολογίζοντας την κλίση για κάθε mini-batch σε κάθε επανάληψη.

Η BGD είναι μια καλή επιλογή για σύνολα δεδομένων μικρού έως μέτριου μεγέθους και όταν χρειάζονται ακριβείς κλίσεις, αν και μπορεί να είναι πιο αργή για μεγαλύτερα σύνολα δεδομένων. Η SGD είναι γνωστή για την αποτελεσματικότητά της όταν χειρίζεται μεγάλα σύνολα δεδομένων και την ικανότητά της να ξεπερνά τα τοπικά ελάχιστα. Ωστόσο, είναι σημαντικό να σημειωθεί ότι παρουσιάζει επίσης υψηλή διακύμανση στις ενημερώσεις παραμέτρων. Η Mini-Batch Gradient Descent είναι μια ευρέως χρησιμοποιούμενη μέθοδος βελτιστοποίησης σε πρακτικές εφαρμογές, ιδίως για εργασίες βαθιάς μάθησης. Συνδυάζει τα πλεονεκτήματα τόσο της BGD όσο και της SGD, καθιστώντας την ιδιαίτερα αποτελεσματική. Το μέγεθος της Mini-Batch παίζει κρίσιμο ρόλο ως υπερπαραμέτρος που πρέπει να συντονιστεί προσεκτικά.

Στην εφαρμογή μας θα χρησιμοποιήσουμε την μεθοδολογία του Mini-Batch.

3.5.1 Vanishing Gradients

Το πρόβλημα του Vanishing Gradient είναι όταν σε Πολυστρωματικά Δίκτυα κατά την διάρκεια του Back Propagation καθώς το σφάλμα του δικτύου διαδίδεται προς τα πίσω από το στρώμα εξόδου στο στρώμα εισόδου, οι παράγωγοι του Error Function σε σχέση με τα βάρη μπορεί να γίνουν εξαιρετικά μικρές. Όταν οι κλίσεις πλησιάζουν το μηδέν, τα βάρη του δικτύου ενημερώνονται ελάχιστα και η μάθηση ουσιαστικά σταματά. Αν τα gradients είναι σταθερά μικρά, μπορεί να εξαφανιστούν μέχρι να φτάσουν στα πρώτα στρώματα, με αποτέλεσμα τα στρώματα αυτά να μαθαίνουν πολύ αργά ή και καθόλου. Αυτό μπορεί να εμποδίσει τη συνολική εκπαίδευση του δικτύου, καθιστώντας δύσκολη τη σύλληψη σύνθετων μοτίβων στα δεδομένα.



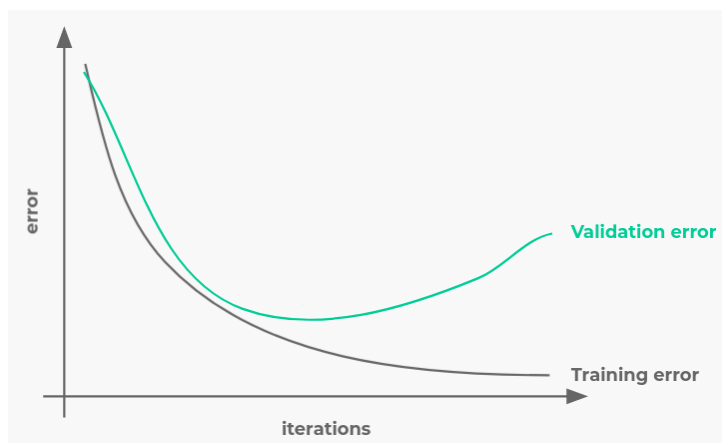
Εικόνα 26: Η σιγμοειδής συνάρτηση και η παράγωγος της

Όπως φαίνεται και στην Εικόνα 26 η οποία πάρθηκε από εδώ³² όταν οι τιμές x γίνονται μεγάλες η μικρές η παράγωγος της σιγμοειδής συνάρτησης αγγίζει το 0, εκεί προκύπτει το πρόβλημα ότι έπειτα από μια σειρά παραγώγων και τιμών του x που είναι μεγάλες τα βάρη γίνονται 0 και η μάθηση σταματά. Για αυτό η σιγμοειδής συνάρτηση δεν χρησιμοποιείται για νευρωνικά δίκτυα πολλαπλών στρωμάτων αφού συμπιέζει ένα μεγάλο χώρο εισόδου σε ένα μικρό χώρο εξόδου μεταξύ 0 και 1. Επομένως, μια μεγάλη αλλαγή στην είσοδο της σιγμοειδούς συνάρτησης θα προκαλέσει μια μικρή αλλαγή στην έξοδο. Ως εκ τούτου, η παράγωγος γίνεται μικρή.

3.6 Το Πρόβλημα της Υπερπροσαρμογής

Ένα πρόβλημα που προκύπτει όταν εκπαιδεύουμε ένα μοντέλο Τεχνητής Νοημοσύνης είναι το πρόβλημα του **Overfitting**. Αυτό που συμβαίνει είναι ότι εκπαιδεύοντας ένα μοντέλο με δεδομένα εκπαίδευσης το μοντέλο γίνεται πάρα πολύ καλό στο να κατηγοριοποιεί συγκεκριμένα τα δεδομένα πάνω στα οποία εκπαιδεύεται αλλά αποτυγχάνει να γενικεύσει την προσέγγιση του σε δεδομένα που δεν έχει ξαναδεί. Η εκδήλωση του Overfitting μπορεί να παρατηρηθεί καθώς το μοντέλο προσαρμόζεται υπερβολικά στις ιδιαιτερότητες των δεδομένων εκπαίδευσης, καταγράφοντας λεπτομέρειες που είναι ουσιαστικά τυχαίες ή μη αντιπροσωπευτικές των ευρύτερων προτύπων των δεδομένων. Το φαινόμενο αυτό είναι ιδιαίτερα έντονο στα βαθιά νευρωνικά δίκτυα, τα οποία διαθέτουν μεγάλο αριθμό παραμέτρων, επιτρέποντάς τους να μαθαίνουν περίπλοκα χαρακτηριστικά και σχέσεις. Στην Εικόνα 27 φαίνεται το πρόβλημα του Overfitting. Όπως διακρίνουμε η πράσινη γραμμή αφορά τα test/validation data και η μαύρη τα training data, όσο προχωράνε τα iteration ή αλλιώς epochs μειώνεται το Error στα Training Data αλλά αυξάνεται το Error στα Validation Data.

³²<https://towardsdatascience.com/the-vanishing-gradient-problem-69bf08b15484>



Εικόνα 27: Overfitting Error

(<https://quantdare.com/mitigatingoverfittingneuralnetworks/>)

”Η κεντρική πρόκληση στη μηχανική μάθηση είναι ότι πρέπει να αποδίδουμε καλά σε νέες, προηγουμένως αθέατες εισόδους - όχι μόνο σε αυτές στις οποίες εκπαιδεύτηκε το μοντέλο μας. Η ικανότητα να αποδίδουμε καλά σε εισόδους που δεν έχουν παρατηρηθεί προηγουμένως ονομάζεται **Generalization**.” [45].

Θα παρουσιάσουμε 5 τρόπους να αποφύγουμε το φαινόμενο του Overfitting από τους οποίους τους 2 τελευταίους θα τους υλοποιήσουμε στην εφαρμογή μας.

Simpler Architecture

Σε αυτή την προσέγγιση θέλουμε να απλοποιήσουμε την αρχιτεκτονική του Νευρωνικού Δικτύου, έτσι αυτό που μπορούμε να κάνουμε είναι να αφαιρέσουμε Layers ή και Νευρώνες έτσι ώστε το Δίκτυο να μικρύνει. Λόγω της φύσης των μοντέλων βαθιάς μάθησης ένα δίκτυο με πολλά στρώματα και πολλούς νευρώνες έχει την ικανότητα να εξειδικεύεται στα δεδομένα που του παρουσιάζονται και να γίνεται πλήρως ικανό να αναγνωρίζει επιτυχώς patterns πάνω στα οποία έχει εκπαιδευτεί, όμως έτσι προκύπτει και το πρόβλημα του Overfitting αφού έχει εξειδικευτεί υπερβολικά πάνω στα training δεδομένα που αποτυγχάνει να κάνει generalize σε δεδομένα που δεν έχει ξαναδεί. Αφαιρώντας Layers και Neurons χάνει την δυνατότητα της υπέρ-εξειδίκευσης και επιτυγχάνει ένα καλύτερο generalize.

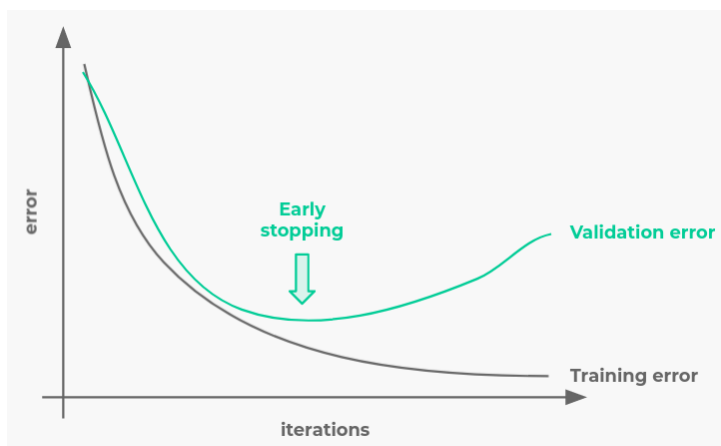
Data Transform

Ιδανικά αυτό που θέλουμε είναι να έχουμε πολλά δεδομένα εκπαίδευσης, αναγκαστικά ένα μοντέλο που έχει λίγα δεδομένα πάνω στα οποία μπορεί να εκπαιδευτεί είναι χειρότερο από ένα που έχει δεκάδες χιλιάδες data. Όμως πολλές φορές δεν έχουμε την δυνατότητα στο να έχουμε στην διάθεση μας ένα τόσο μεγάλο Data Set, αυτό που μπορούμε να κάνουμε είναι να επεξεργαστούμε με μεθόδους transformation τα δεδομένα που ήδη έχουμε και να τα παρουσιάζουμε σαν καινούρια. Χαρακτηριστικό της προσέγγισης είναι ότι αλλάζει το τραγούδι χωρίς όμως να αλλάζει το είδος στο οποίο ανήκει. Συγκεκριμένα σε αρχεία ήχου με τα οποία ασχολούμαστε σε αυτή την εφαρμογή αναφέρονται μεθοδολογίες στο [46], μερικές είναι οι παρακάτω:

- **Pitch Shifting:** Να ανεβάσουμε/κατεβάσουμε την τονικότητα ενός κομματιού, δηλαδή από μια κλίμακα να το μετατρέψουμε σε μια άλλη, κι έτσι παράγεται το ίδιο τραγούδι αλλά σε διαφορετική κλίμακα που πρακτικά είναι διαφορετικό και το μοντέλο επεκτείνει την ικανότητα του να αναγνωρίζει τραγούδια σε διαφορετικές κλίμακες.
- **Time Stretching:** Να αλλάξουμε το bpm τον ρυθμό δηλαδή ενός τραγουδιού, έτσι το τραγούδι γίνεται πιο αργό ή πιο γρήγορο και όπως και με το Pitch Shifting είναι ένα νέο τραγούδι.
- **Noise Addition:** Η προσθήκη θορύβου σε κάποιο τραγούδι.

Early Stop

Μια από τις γνωστές μεθοδολογίες για να αποτραπεί το Overfitting. Όπως μπορούμε να διακρίνουμε στην Εικόνα 27 από ένα σημείο των epochs και μετά αρχίζει και δημιουργείται μια διαφορά στο error του training data και του test data το οποίο όσο προχωράει και αυξάνεται, εν ολίγη με κάθε epoch το μοντέλο γίνεται χειρότερο στο generalization. Ο αλγόριθμος του Early Stopping σταματάει την εκπαίδευση του μοντέλου Βαθιάς Μάθησης όταν βλέπει ότι το error μεταξύ training και test data δεν βελτιώνεται. Ένα σημείο του Early Stopping φαίνεται στην Εικόνα 29 όπου το βέλος δείχνει που θα σταματήσει η μάθηση.

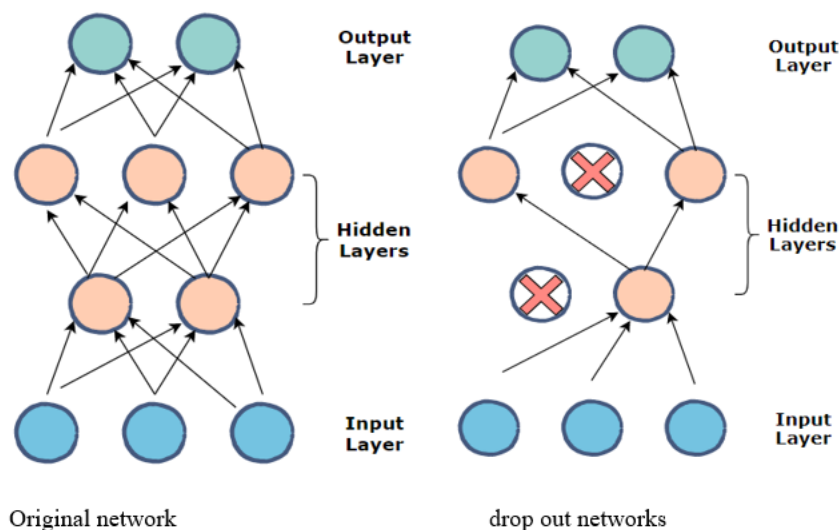


Εικόνα 28: Early Stop

Dropout

Αυτή η μεθοδολογία μαζί με την επόμενη που θα αναφέρουμε είναι αυτές που θα υλοποιήσουμε στην εφαρμογή μας. Αυτό που γίνεται στην Dropout είναι ότι κατά την φάση της εκπαίδευσης επιλέγονται τυχαία ένας αριθμός από νευρώνες και προσωρινά απενεργοποιούνται, στην συνέχεια ενεργοποιούνται και απενεργοποιούνται άλλοι. Η προσέγγιση θεωρητικά δημιουργεί έναν μέσο όρο για το μοντέλο. Εκπαιδύοντας πολλαπλά υποδίκτυα με διαφορετικά σύνολα νευρώνων που απενεργοποιούνται, το δίκτυο μαθαίνει ουσιαστικά να είναι πιο ανθεκτικό και εξαρτάται λιγότερο από οποιοδήποτε συγκεκριμένο σύνολο βαρών. Επίσης όταν οι νευρώνες σε ένα δίκτυο βασίζονται υπερβολικά στην παρουσία άλλων συγκεκριμένων νευρώνων προκύπτει το πρόβλημα του **co-adaptation**. Το dropout βοηθά στην αποτροπή αυτού του φαινομένου, καθιστώντας πιο δύσκολη την εξειδίκευση των νευρώνων στην αναγνώριση ορισμένων χαρακτηριστικών, αναγκάζοντάς τους να είναι πιο γενικά χρήσιμοι. Ο

κάθε νευρώνας έχει μια συγκεκριμένη πιθανότητα να κάνει dropout και αυτό ορίζεται από την μεταβλητή dropout probability η οποία λαμβάνει τιμές συνήθως στο εύρος $[0.1, \dots, 0.5]$.



Εικόνα 29: Dropout Method

(<https://medium.com>)

Regularization

Σε αυτή την τεχνική προσθέτουμε ένα **penalty** στην Error Function. Ουσιαστικά αυτό που θέλουμε να κάνουμε είναι να τιμωρούνται τα μεγάλα βάρη στους νευρώνες, αυτή η ποινή αποθαρρύνει το μοντέλο από το να γίνει υπερβολικά πολύπλοκο ή να έχει μεγάλες τιμές παραμέτρων. Οι δύο κύριοι τύποι Regularization στη βαθιά μάθηση είναι η L1 και η L2. Αυτές ενημερώνουν τη γενική συνάρτηση κόστους με την προσθήκη ενός άλλου όρου, γνωστού ως όρος κανονικοποίησης (*regularization term*) που συμβολίζεται με λ , όσο πιο υψηλό το λ τόσο πιο υψηλή και η ποινή που ασκούμε στο δίκτυο.

$$\text{Συνάρτηση Κόστους: } E(p, y) = \frac{1}{2}(p - y)^2$$

$$\text{L1: } E(p, y) = \frac{1}{2}(p - y)^2 + \lambda \sum |W_i|$$

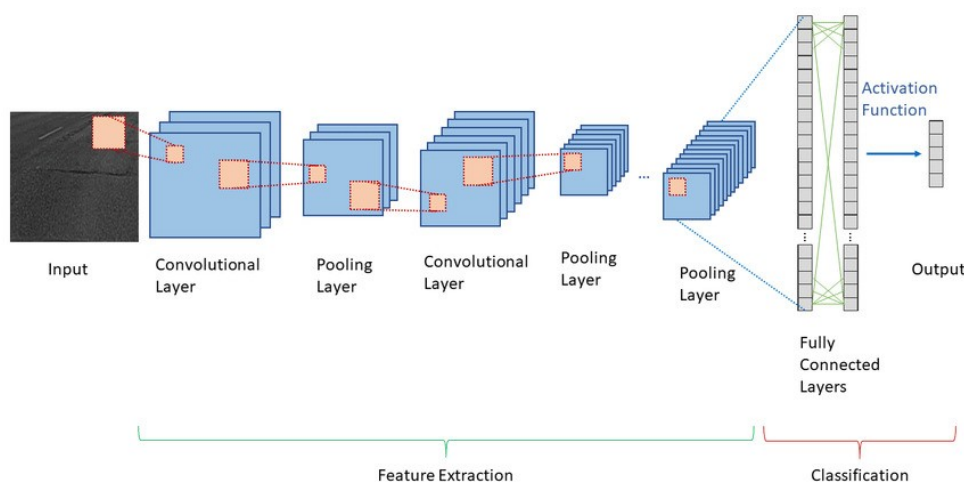
$$\text{L2: } E(p, y) = \frac{1}{2}(p - y)^2 + \lambda \sum |W_i|^2$$

Η L1 προσέγγιση, επίσης γνωστή ως νόρμα L1 ή Lasso, καταπολεμά το Overfitting συρρικνώνοντας τις παραμέτρους προς το 0. Αυτό καθιστά ορισμένα χαρακτηριστικά παρωχημένα. Πρόκειται για μια μορφή επιλογής χαρακτηριστικών, διότι όταν αναθέτουμε σε ένα

χαρακτηριστικό με βάρος 0, πολλαπλασιάζουμε τις τιμές των χαρακτηριστικών με το 0, το οποίο επιστρέφει 0, εξαλείφοντας τη σημασία αυτού του χαρακτηριστικού. Εάν τα χαρακτηριστικά εισόδου του μοντέλου μας έχουν βάρη πιο κοντά στο 0, η νόρμα L1 θα ήταν αραιή. Μια επιλογή των χαρακτηριστικών εισόδου θα είχε βάρη ίσα με το μηδέν και τα υπόλοιπα θα ήταν μη μηδενικά. Έτσι το L1 προτιμάται όταν η αραιότητα των χαρακτηριστικών και η επιλογή είναι σημαντικές, καθώς τείνει να οδηγεί ορισμένα βάρη στο μηδέν ακριβώς.

Η L2 (Ridge) είναι κατάλληλη για τον έλεγχο των συνολικών μεγεθών βάρους και την αποφυγή ακραίων τιμών, παρέχοντας ισορροπία μεταξύ των χαρακτηριστικών, καταπολεμά το overfitting αναγκάζοντας τα βάρη να είναι μικρά, αλλά χωρίς να τα κάνει ακριβώς 0. Η κανονικοποίηση L2 δεν εκτελεί επιλογή χαρακτηριστικών, αφού τα βάρη μειώνονται μόνο σε τιμές κοντά στο 0 αντί για 0. Η κανονικοποίηση L1 έχει ενσωματωμένη επιλογή χαρακτηριστικών. Η κανονικοποίηση L1 είναι ανθεκτική στις ακραίες τιμές, ενώ η κανονικοποίηση L2 όχι. Η L2 έχει την δυνατότητα να μαθαίνει πιο περίπλοκα patterns. Στην εφαρμογή μας που τα δεδομένα είναι σχετικά σύνθετα θα χρησιμοποιήσουμε την L2 προσέγγιση.

3.7 Convolutional Neural Networks



Εικόνα 30: Αρχιτεκτονική ενός CNN

<https://www.linkedin.com/in/nourhan-moustafa/>

Στην Εικόνα 30 φαίνεται η αρχιτεκτονική ενός Συλλεκτικού Νευρωνικού Δικτύου (CNN). Στο δεξί μέρος της εικόνας που αναφέρονται τα fully connected layers είναι το μέρος του νευρωνικού δικτύου που είναι ίδιο με το MLP που αναλύσαμε στα προηγούμενα υποκεφάλαια, σε αυτό το οποίο διαφοροποιείται το CNN από το MLP είναι τα προηγούμενα στάδια επεξεργασίας από τα οποία περνούν τα δεδομένα.

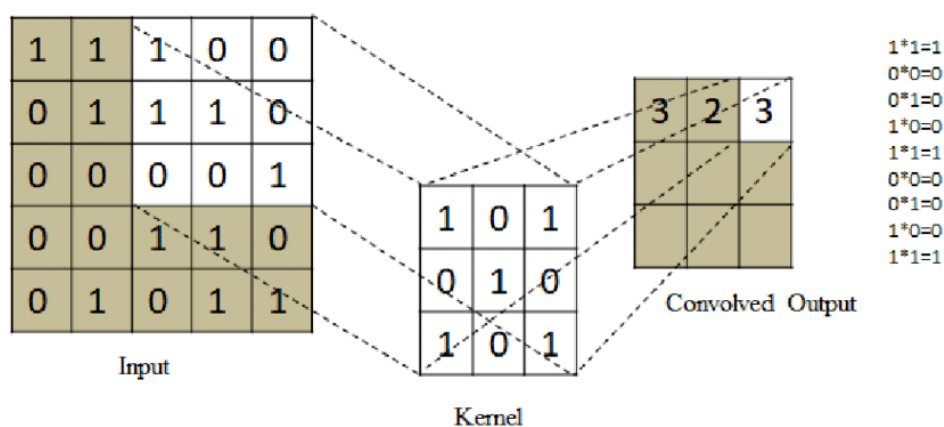
Πιο συγκεκριμένα το CNN δουλεύει με πίνακες από αριθμούς, μια εικόνα είναι ένας δι-διάστατος πίνακας από αριθμούς όπου το κάθε κελί αναπαριστά ένα pixel και παίρνει τιμές από 0-255 για ασπρόμαυρες εικόνες όπου το 0 είναι το μαύρο, 255 το λευκό και οι ενδιάμεσες είναι όλα τα ενδιάμεσα γκρι ή μπορεί το κελί να διαθέτει τρεις τιμές (x, y, z) οι οποίες

αναφέρονται στο R,G,B σύστημα. Ως συμπέρασμα από το παραπάνω το CNN κυρίως χρησιμοποιείται για Digital Image Processing (DIP) και τεχνικές του DIP εφαρμόζονται για να ενισχύσουν την αποδοτικότητα του CNN.

Πως συνδέονται όμως οι Εικόνες με την Μουσική, εξάλλου η Μουσική είναι ήχος όχι εικόνα. Στο κεφάλαιο 4.4 δείξαμε πως από τον ήχο μεταβαίνουμε σε εικονική αναπαράσταση όπου αυτή η αναπαράσταση είναι τα MFCC. Υπάρχουν κι άλλες αναπαραστάσεις αλλά εμείς δουλεύουμε κυρίως με τα MFCC, σε αυτήν την φάση ο ήχος έχει γίνει εικόνα και αυτή η εικόνα θα είναι τα δεδομένα εισόδου του CNN που θα δημιουργήσουμε.

3.7.1 Convolution

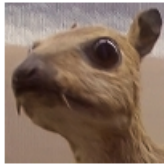
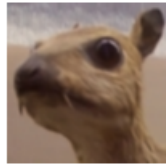
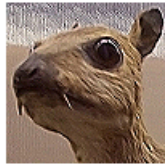

Το Convolution ή αλλιώς Συνέλιξη είναι η βασική πράξη που πραγματοποιείται στο Convolution Layer όπως φαίνεται στην εικόνα 30. Αυτό που κάνει η συνέλιξη είναι μέσω του πολλαπλασιασμού ενός πίνακα **Kernel** και της Εικόνας εισόδου να δημιουργήσει έναν νέο **πίνακα** ο οποίος θα περιέχει πιο ουσιώδεις πληροφορίες από αυτόν που περιείχε ο αρχικός πίνακας-εικόνα. Όπως φαίνεται στην Εικόνα 31 κάθε κελί της αρχικής εικόνας (Input) πολλαπλασιάζεται κάθε κελί του Kernel και αφού αθροιστούν οι πολλαπλασιασμοί το αποτέλεσμα αποθηκεύεται στον νέο πίνακα Convolved Output, στην συνέχεια το παράθυρο kernel "κυλάει" πάνω στην εικόνα και εκτελεί την επόμενη συνέλιξη.



Εικόνα 31: Η πράξη της Συνέλιξης
<https://www.researchgate.net/>

Οι αριθμοί οι οποίοι εμπεριέχει ένας kernel πίνακας δεν είναι τυχαίοι, εξυπηρετούν τον σκοπό για τον οποίο προορίζεται ο αλγόριθμος να επιτελέσει. Παραδείγματα φίλτρων/kernels φαίνονται στη Εικόνα 32.

Στην επιστήμη του DIP διάφοροι Kernels χρησιμοποιούνται στην Εικόνα σειριακά για να εξαχθούν χρήσιμες πληροφορίες, για παράδειγμα στην αναγνώριση προσώπου από την εικόνα εισόδου θα πρώτα θα εφαρμοστεί ένας Kernel ο οποίος θα αφαιρεί το χρώμα από την φωτογραφία για απλότητα στην επεξεργασία, έπειτα ένας άλλος kernel θα αφαιρεί τον θόρυβο από την φωτογραφία, έπειτα θα εφαρμοστεί ένας Kernel που θα δίνει έμφαση στις γωνίες, μετά από τις γωνίες θα εφαρμοστεί ένας Kernel που θα ρυθμίσει την κατανομή της έντασης της εικόνας για να βελτιώσει την ορατότητα των χαρακτηριστικών του προσώπου και ούτω

<i>Original</i>	<i>Gaussian Blur</i>	<i>Sharpen</i>	<i>Edge Detection</i>
$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$
			

Εικόνα 32: Διάφοροι τύποι Kernels
<https://medium.com/analytics-vidhya/>

καθεξής.

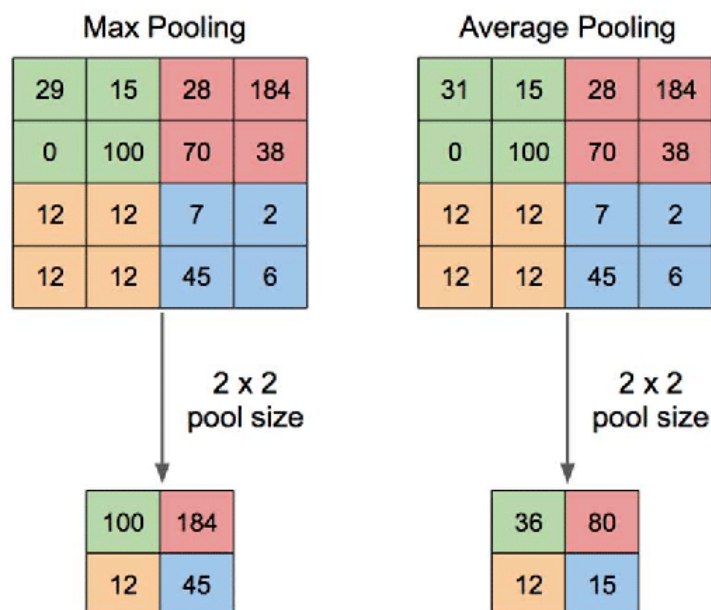
Σε ένα τέτοιο παράδειγμα μπορούμε να δημιουργήσουμε ένα Pipeline από Kernel επεξεργασίες οι οποίες θα προετοιμάσουν τα δεδομένα για να εισαχθούν στο νευρωνικό δίκτυο και να εκπαιδευτεί. Ωστόσο τι κάνουμε αν δεν ξέρουμε ποια χαρακτηριστικά να εξάγουμε; Τι χαρακτηριστικά είναι τα πιο χρήσιμα να εξαχθούν από ένα μουσικό κομμάτι; Δεν γνωρίζουμε ακριβώς, και να γνωρίζαμε σίγουρα θα υπήρχαν χαρακτηριστικά που αλλάζουν από κομμάτι σε κομμάτι. Την απάντηση σε αυτό το ερώτημα μας την δίνουν τα CNN όπου μέσω του Back Propagation που έχουν δημιουργούν Kernels με μορφή που καλύτερα εξάγει την πληροφορία η οποία ανεβάζει την απόδοση του Δικτύου. Ουσιαστικά αυτά τα νευρωνικά δίκτυα μαθαίνουν μόνα τους ποια χαρακτηριστικά να εξάγουν από μια αναπαράσταση Εικόνας μέσω του Back Propagation το οποίο όπως είδαμε και στο κεφάλαιο 3.4 από μόνο του αλλάζει τις τιμές των βαρών ώστε το Error να μειώνεται, στην δική μας περίπτωση τώρα στα CNN τα βάρη είναι οι πίνακες Kernel.

3.7.2 Pooling

Αφού ολοκληρώσουμε την διαδικασία του Convolution περνάμε στην φάση του Pooling. Το Pooling δεν είναι τίποτα άλλο πέρα από την μείωση των διαστάσεων της Εικόνας για την επιτάχυνση των υπολογισμών και την απλοποίηση των πράξεων. Αυτό γίνεται πάλι μέσω ενός Kernel ο οποίος από μια περιοχή pixel επιλέγει το μεγαλύτερο (**Max-Pooling**) ή εκτελεί έναν μέσο όρο των γειτονικών pixel **Average Pooling**. Υπάρχουν κι άλλοι μεθόδοι Pooling αλλά στα CNN επιλέγονται κυρίως οι δύο που αναφέραμε. Στην Εικόνα 33 φαίνονται οι δύο προσεγγίσεις.

3.7.3 Μετάβαση σε Fully-Connected Layer

Μετά από μια σειρά Convolution και Pooling η εικόνα θα έχει μειωθεί αρκετά και μόνο οι πληροφορίες που μας ενδιαφέρουν θα έχουν παραμείνει, σε αυτό το σημείο αυτό που κάνουμε είναι την εικόνα να την κάνουμε "επίπεδη" ή αλλιώς flatten δηλαδή να την μετατρέψουμε σε διάνυσμα έτσι ώστε να εισαχθεί στο πρώτο Layer νευρώνων και από εκεί να ξεκινήσει



Εικόνα 33: Max και Average Pooling

<https://towardsai.net/p/l/>

το forward propagation. Για παράδειγμα αν μια εικόνα 1028x1028 έχει φτάσει σε διαστάσεις 64x64 τότε το flatten θα την μετατρέψει σε ένα διάνυσμα 4096x1, αυτό σημαίνει πως πρέπει στο layer εισόδου να υπάρχουν 4096 νευρώνες για να υποδεχθούν τις τιμές. Από όσο καταλαβαίνουμε όσο περισσότερο μειώνουμε την εικόνα τόσο λιγότερους νευρώνες χρειαζόμαστε, έτσι βελτιώνεται η πολυπλοκότητα του δικτύου και η ταχύτητα του. Από εκεί και έπειτα η διαδικασία είναι όπως είπαμε ίδια με την διαδικασία σε ένα NLP.

3.8 Recurrent Neural Networks

Τα αναδρομικά νευρωνικά δίκτυα (RNN) έρχονται για να λύσουν προβλήματα στα οποία έχουν σημασία όχι μόνο τα δεδομένα σε μια συγκεκριμένη χρονική στιγμή, αλλά και τα δεδομένα που προηγήθηκαν από αυτή την στιγμή. Μπορούμε να σκεφτούμε ένα παράδειγμα τέτοιας ανάγκης όταν δούμε μια φωτογραφία μιας μπάλας που κινείται και να χρειαστεί να προβλέψουμε προς τα που θα κινηθεί. Από ένα μόνο στιγμιότυπο δεν θα μπορούσαμε ποτέ να συμπεράνουμε την κατεύθυνση της μπάλας, εκτός αν μας δωθεί μια ακόμη φωτογραφία που δείχνει που βρισκόταν η μπάλα πριν το μετέπειτα στιγμιότυπο που μας δόθηκε, από αυτά τα δύο δεδομένα θα μπορούμε να εξάγουμε το συμπέρασμα της κίνησης της μπάλας.

Κάπως έτσι τα RNN βρίσκουν γιγάντια χρήση στους τομείς του NLP όπου έχει σημασία η σειρά των λέξεων μέσα σε μια πρόταση για την εξαγωγή συμπεράσματος³³, Speech Recognition, Time Series Prediction όπου προβλέπονται μελλοντικές τιμές σε δεδομένα χρονοσειρών όπως τιμές μετοχών, καιρικές συνθήκες, κατανάλωση ενέργειας και πολλά άλλα.

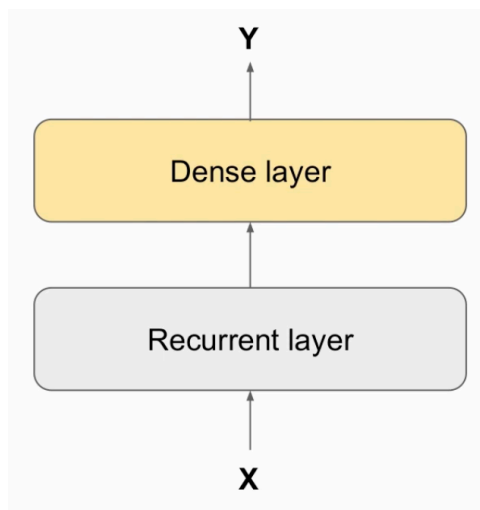
Στον κλάδο της Μουσικής που ασχολούμαστε η έννοια της σειράς έχει σημασία αφού το κλειδί που θα χρησιμοποιηθεί, το μέτρο στο οποίο θα ολοκληρώσει ένας κύκλος, το αν η μουσική δυναμώνει η χαμηλώνει έχει μια λογική σειρά στον χρόνο και η κάθε στιγμή προκύπτει

³³Π.χ. άλλη σημασία έχει η πρόταση "John loves Mary" και άλλη σημασία το "Mary loves John"

από την προηγούμενη της, έτσι τα RNN έχουν χρήση σε αυτόν τον τομέα.

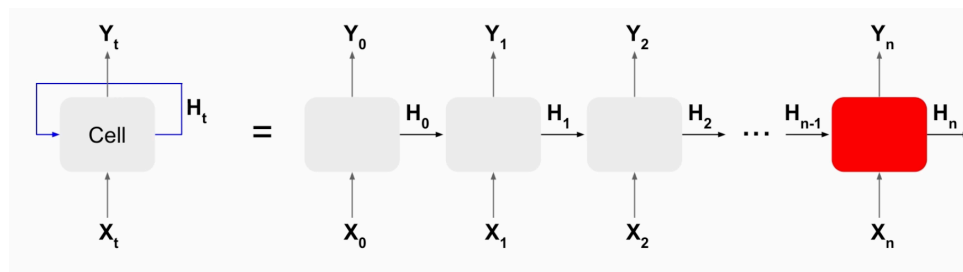
3.8.1 Αρχιτεκτονική

Όπως βλέπουμε στην Εικόνα 34 ένα RNN αποτελείται από ένα Αναδρομικό Στρώμα όπου εκεί είναι που εστιάζουμε την προσοχή και εκεί βρίσκεται η ουσία του RNN και ένα Dense στρώμα στο οποίο ακολουθείται η ίδια λογική με το Multilayer Perceptron.



Εικόνα 34: Αρχιτεκτονική ενός RNN
<https://github.com/musikalkemist>

Το πιο βασικό κομμάτι των RNN's ονομάζεται Πυρήνας Cell. Ο πυρήνας είναι μια υπολογιστική μονάδα όπως φαίνεται στην εικόνα 35 όπου το κάθε είδος RNN διαφέρει από το άλλο βάση των τροποποιήσεων που υπάρχουν μέσα στον πυρήνα, έτσι υπάρχουν πολλά είδη πυρήνων. Βλέπουμε ότι εισάγεται στον πυρήνα η είσοδος X_t γίνεται μια επεξεργασία η οποία εξαγεί το H_t και Y_t και το H_t χρησιμοποιείται σαν ανατροφοδότηση στον πυρήνα για να εξαχθεί η νέα τιμή. Έτσι για παράδειγμα όταν εισάγουμε το δεδομένο X_t για να εξαχθεί το συμπέρασμα η είσοδος X_t θα επεξεργασθεί με την χρήση του H_{t-1} για να εξαχθεί το Y_t . Κάθε φορά εισάγονται νέα δεδομένα και από είσοδο και από προηγούμενη κατάσταση αλλά ο πυρήνας παραμένει ίδιος.

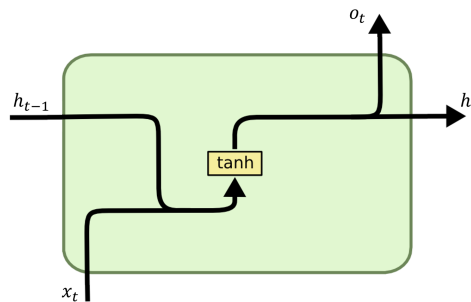


Εικόνα 35: Ο πυρήνας του RNN
<https://github.com/musikalkemist>

Στην Εικόνα 36 φαίνεται το εσωτερικό του πυρήνα του RNN που είναι η πιο απλή μορφή που μπορεί να έχει. Από το σχήμα βλέπουμε ότι η είσοδος x_t προστίθεται με την προηγούμενη κατάσταση h_{t-1} και το αποτέλεσμα περνάει μέσα από μια συνάρτηση ενεργοποίησης \tanh

και εξάγεται ως αποτέλεσμα o_t όπως και οδηγείται στον επόμενο πυρήνα ως h_t . Αν υπάρχει βάρος U στο x_t και βάρος W στο h_{t-1} τότε η μαθηματική φόρμουλα για να βρούμε το h_t θα είναι η 22

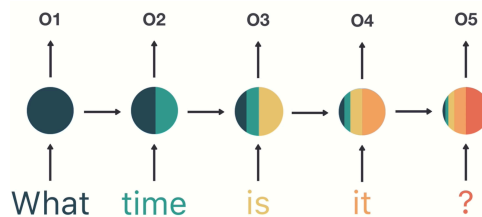
$$h_t = \tanh(Ux_t + Wh_{t-1}) \tag{22}$$



Εικόνα 36: Ο πιο απλός πυρήνας RNN
<https://medium.com/@imjeremyhi/>

Ο λόγος που χρησιμοποιούμε την συνάρτηση ενεργοποίησης **tanh** είναι γιατί η **tanh(x)** λαμβάνει τιμές $[-1, 1] \forall x \in \mathbb{R}$, έτσι αποφεύγουμε το πρόβλημα του **exploding gradient** όπου οι τιμές τείνουν να "εκρήγνυνται" προς πολύ μεγάλες τιμές όπως και το πρόβλημα του vanishing gradients.

Τα προβλήματα που προκύπτουν από τα απλά RNN είναι ότι δεν υπάρχει μνήμη μακράς διάρκειας, οι τιμές που προωθούνται από μια προηγούμενη κατάσταση σε μια επόμενη κατάσταση είναι οι πιο πρόσφατες. Όπως φαίνεται στην Εικόνα 37 η πιο πρόσφατη είσοδος και έξοδος προηγούμενης κατάστασης έχει πολύ μεγαλύτερη βαρύτητα από την πιο παλιά h . Έτσι όπως φαίνεται στο παράδειγμα το ερωτηματικό στο τέλος καταλαμβάνει πολύ μεγαλύτερο χώρο και σημαντικότητα στην επεξεργασία της εξόδου από την λέξη "When" που είναι πολύ σημαντικό.

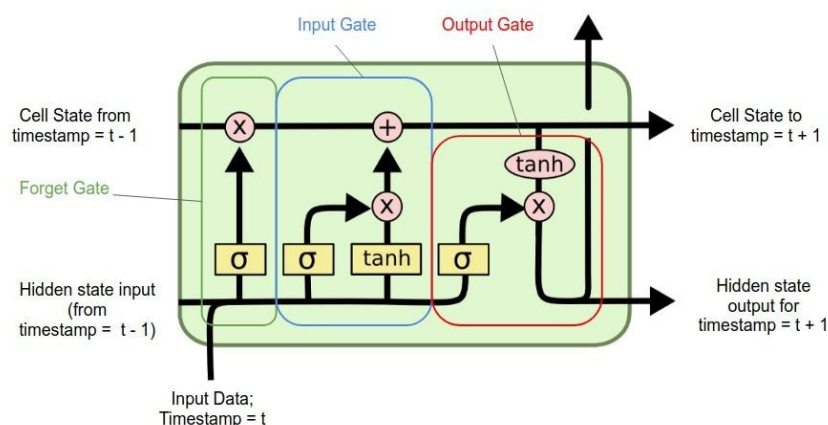


Εικόνα 37: Χάσιμο παλιές πληροφορίας από RNN
<https://towardsdatascience.com/>

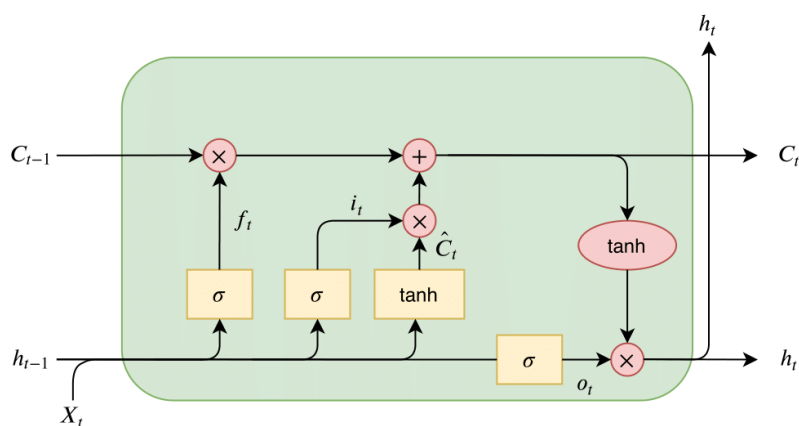
Έτσι το δίκτυο δεν μπορεί να μάθει από το μακρινό παρελθόν και δεν μπορεί να βρρίσκει μοτίβα με μακροχρόνιες εξαρτήσεις το οποίο είναι κάτι σημαντικό για τα αρχεία ήχου με τα οποία ασχολούμαστε γιατί ένα κομμάτι εξαρτάται από τις παρελθοντικές καταστάσεις από τις οποίες έχει περάσει.

3.8.2 LSTM

Το παραπάνω πρόβλημα έρχονται να λύσουν τα LSTM - Long Short Term Memory δίκτυα. Τα LSTM είναι ένας ειδικός τύπος RNN, έχουν το χαρακτηριστικό ότι μπορούν και μαθαίνουν από μακροχρόνια μοτίβα, αναγνωρίζουν μοτίβα της έκτασης των 100 βημάτων αλλά πάλι δυσκολεύονται με μοτίβα πολλών εκατοντάδων βημάτων και πάνω. Η διαφορά που έχουν από τα RNN είναι στην πυρήνα που διαθέτουν όπως φαίνεται στην Εικόνα 38.



Εικόνα 38: Πυρήνας του LSTM
<https://www.linkedin.com/pulse>

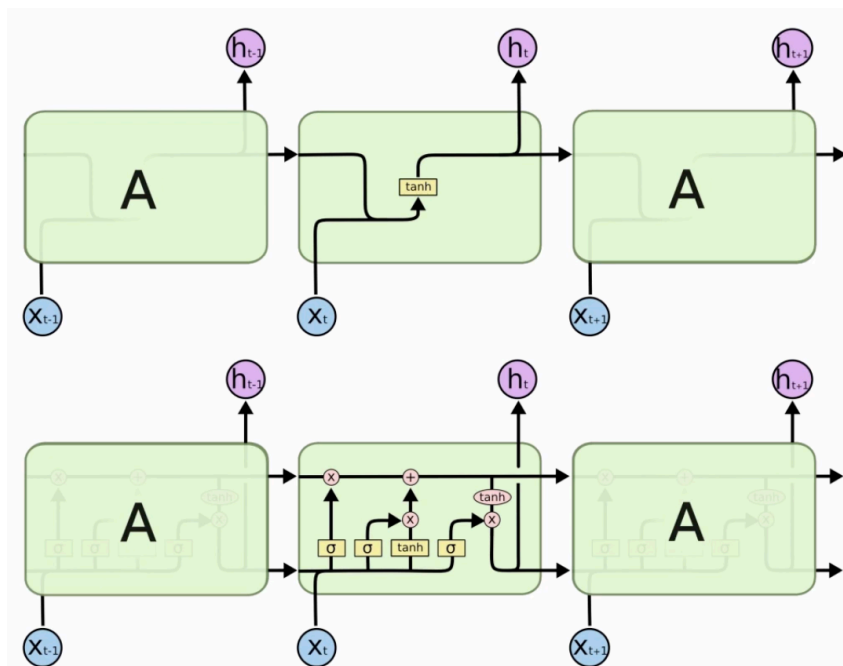


Εικόνα 39: Άλλη αναπαράσταση του LSTM
https://thorimar.com/post/insight_into_lstm/

Θα αναφερθούμε σύντομα στην λειτουργία του. Τα LSTM λειτουργούν με βάση τα παρακάτω.

- **Cell State.**

Το Cell State C_t είναι υπεύθυνο για την μακρόχρονη πληροφορία που μεταβιβάζεται από πυρήνα σε πυρήνα. Οι επεξεργασίες στο Cell State είναι περισσότερες και αφορούν πρώτα την πληροφορία που θα παραμείνει μέσω του Forget Gate και την πληροφορία που θα εισαχθεί μέσω του Input Gate. Οι επεξεργασίες που εφαρμόζονται στο Cell State το καθιστούν και πιο σταθερό σε ότι έχει να κάνει με τις τιμές των gradients του το οποίο είναι το βασικό πρόβλημα της εκπαίδευσης των RNN.



Εικόνα 40: Σύγκριση RNN (πάνω) με LSTM (κάτω)

- **Hidden State.**

Το Hidden State h_t είναι υπεύθυνο για την βραχυχρόνια πληροφορία που αποθηκεύεται και εξάγεται από τον πυρήνα. Αφορά την πιο πρόσφατη κατάσταση των προηγούμενων πυρήνων μαζί με την είσοδο που εισάγεται και έχει λιγότερες επεξεργασίες.

- **Forget Gate.**

Σε αυτή την πύλη όπως φαίνεται στην Εικόνα 38 μέσα στο πράσινο περίγραμμα αποφασίζει ποια πληροφορία θα παραμείνει από την προηγούμενη κατάσταση και ποια θα εξαλειφθεί. Αυτό πραγματοποιείται με χρήση της συνάρτησης ενεργοποίησης **sigmoid** (σ). Η $\sigma(x)$ λαμβάνει τιμές $[0, 1]$, έτσι αν κάποια πληροφορία είναι άχρηστη θα εξαλειφθεί πολλαπλασιασμένη με 0 η αν κάποια πληροφορία είναι σημαντική θα παραμείνει πολλαπλασιασμένη με 1. Ο τύπος της Forget Gate είναι ο 23, βλέπουμε ότι η είσοδος προστίθεται στο διάνυσμα με την προηγούμενη κατάσταση h_{t-1} και προστίθεται και ένας όρος bias b_f όπου όλο αυτό περνάει από την sigmoid συνάρτηση ενεργοποίησης.

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \quad (23)$$

- **Input Gate.**

Η Input Gate (μπλε) αποφασίζει ποια νέα πληροφορία θα εισαχθεί στο διάνυσμα, και αυτό το κάνει πολλαπλασιάζοντας την πληροφορία που παραμένει από την sigmoid συνάρτηση ενεργοποίησης με το αποτέλεσμα συμπίεσης μέσω της **tanh**.

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \quad (24)$$

$$C'_t = \tanh(W_c[h_{t-1}, x_t] + b_c) \quad (25)$$

$$C_t^i = C'_t * i_t \quad (26)$$

- **Output Gate.**

Τέλος το Output Gate (κόκκινο) παράγει την τελική έξοδο του πυρήνα όπου το διάνυσμα **Cell State** περνάει μέσα από την συνάρτηση ενεργοποίησης **tanh** όπου συμπιέζεται και το αποτέλεσμα αυτού πολλαπλασιάζεται με το τι παραμένει από τον τύπο 23.

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \quad (27)$$

$$h_t = o_t * \tanh(C_t) \quad (28)$$

Η τελική παράσταση που αφορά του Cell State είναι η 29

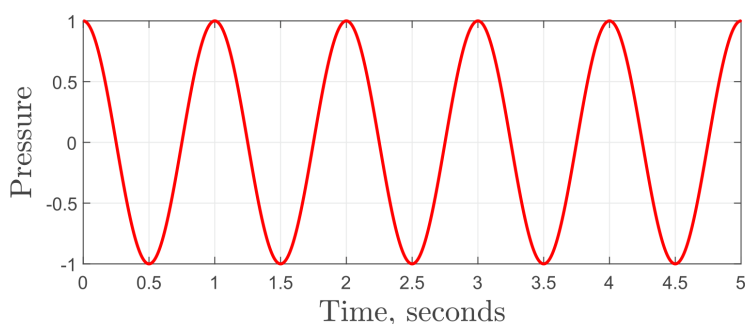
$$C_t = C_t^f + C_t^i \quad (29)$$

4 Ήχος & Μουσική

Κάθε μοντέλο Τεχνητής Νοημοσύνης για να μπορεί να εκπαιδευτεί και να φτάσει σε ένα σημείο όπου με βάση κάποια είσοδο να εξάγει συμπεράσματα ανάλογα το πρόβλημα που καλείται να λύσει χρειάζεται τα κατάλληλα δεδομένα που είναι σχετικά με τον τομέα της εφαρμογής. Για παράδειγμα σε ένα πρόβλημα που αφορά Image Recognition τα δεδομένα που εκπαιδεύεται ένα NN είναι *pixels*, στο *NLP* είναι σύνολα δεδομένων κειμένου επισημασμένα με συναισθήματα, σε Ανίχνευση Απάτης είναι Δεδομένα συναλλαγών με χαρακτηρισμένες περιπτώσεις απάτης ή μη απάτης και πολλά άλλα. Σε αυτό το κομμάτι της εργασίας θα δείξουμε τις βασικές γνώσεις που απαιτούνται για τον τύπο δεδομένων που θα δουλέψουμε στην εργασία που είναι τα **Δεδομένα Ήχου**. Πιο συγκεκριμένα θα δούμε έννοιες όπως Ήχος, *Pitch*, Ένταση και πράγματα πιο προχωρημένα όπως *Fourier Transform*, *Spectrograms* και *MFCC's*. Όλα αυτά θα μας χρειαστούν έτσι ώστε το Νευρωνικό Δίκτυο το οποίο θα υλοποιήσουμε να τα δέχεται σαν είσοδο και να βγάζει συμπεράσματα για το στυλ μουσικής που είναι το εκάστοτε κομμάτι.

4.1 Ήχος

Ήχος παράγεται όταν ένα σώμα βρίσκεται σε κατάσταση που δονείται και οι δονήσεις του επηρεάζουν τα μόρια του αέρα και τα κάνουν να ταλαντεύονται. Έτσι δημιουργείται μια εναλλαγή στην πίεση του αέρα που ως αποτέλεσμα δημιουργεί ένα **κύμα**. Ο κύριος τρόπος που αναπαριστούμε ένα τέτοιο κύμα ήχου είναι μέσω των *Waveform* που φαίνεται στην Εικόνα 41.



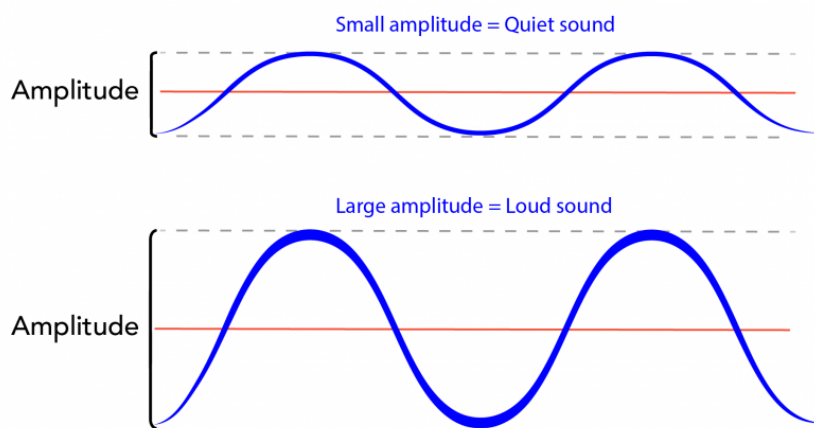
Εικόνα 41: Waveform

Ένα τέτοιο κύμα μπορούμε να το αναπαραστήσουμε χρησιμοποιώντας μια μετρική *Amplitude* ή αλλιώς *Πλάτος* (A) όπου φαίνεται από το "ύψος" που έχει το κύμα που δείχνει την ένταση όπως θα δούμε στην συνέχεια του κύματος και χρόνο (t). Κάποιες άλλες βασικές πληροφορίες ενός κύματος είναι η **περίοδος** (T) που έχει η οποία είναι ο χρόνος που απαιτείται (σε **sec**) για να γίνει μια επανάληψη του κύματος και η **συχνότητα** όπου $f = \frac{1}{T}$. Η μαθηματική αναπαράσταση του κύματος είναι η (30).

$$y(t) = A \sin(2\pi ft + \phi) \quad (30)$$

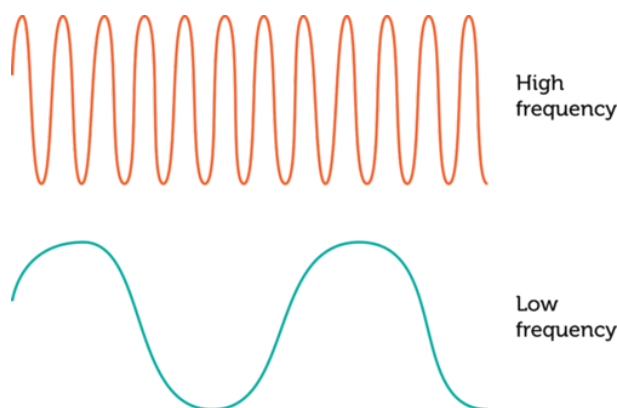
Στον τομέα της Μουσικής που θα ασχοληθούμε οι έννοιες του Πλάτους και της Συχνότητας

συνδέονται με τις έννοιες της Έντασης και του Pitch.



Εικόνα 42: Μεγάλο και Μικρό Πλάτος

Όπως φαίνεται στην Εικόνα 42 ένα κύμα με μικρό πλάτος (το πάνω) είναι ένας ήχος πιο σιγανός από ένα κύμα με μεγαλύτερο πλάτος (το κάτω), δηλαδή όσο πιο δυνατά παίζει κάποιος ένα όργανο τόσο μεγαλύτερο πλάτος θα έχει το κύμα που δημιουργεί.

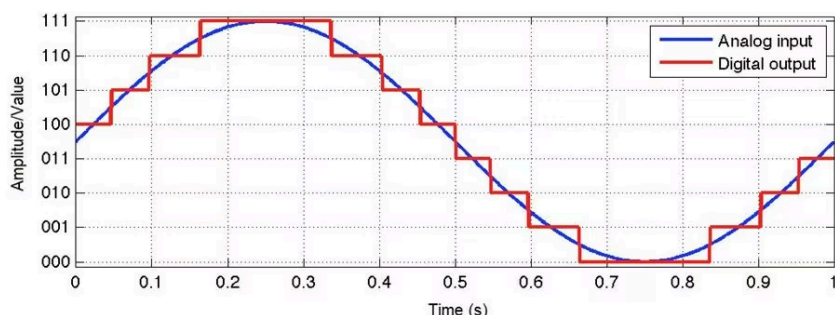


Εικόνα 43: Μεγάλη και Μικρή Συχνότητα

Επίσης όπως φαίνεται στην Εικόνα 43 ένα κύμα με μεγάλη συχνότητα (πάνω) έχει πιο ψηλό pitch ενώ το κύμα με χαμηλότερη συχνότητα (κάτω) έχει πιο χαμηλό pitch. Παράδειγμα ενός οργάνου με υψηλό pitch είναι το βιολί, ενώ ένα όργανο με χαμηλό pitch είναι ένα κοντραμπάσο ή μια τούμπα, γενικότερα υψηλό pitch είναι το Άλτο και χαμηλό pitch είναι το Μπάσο.

Όλα τα όργανα ή και η φωνή μας όταν δημιουργούν κύματα όπως δείξαμε παραπάνω παράγουν συνεχή κύματα, ένα συνεχές κύμα έχει μέσα άπειρες διαβαθμίσεις στο πλάτος και στον χρόνο και είναι μια μορφή που δεν μπορεί να αποθηκευτεί στον υπολογιστή, για αυτό το λόγο ένα κύμα ήχου το μετατρέπουμε σε ψηφιακή μορφή όπως φαίνεται στην Εικόνα 44. Η μπλε γραμμή είναι το συνεχές φυσικό κύμα και η κόκκινη είναι αυτό που παράγουμε μέσω του υπολογιστή. Θέτουμε ένα συγκεκριμένο αριθμό από πλάτη όπως φαίνεται στον άξονα *Amplitude* όπου κάθε φορά ανάγουμε την τιμή του συνεχούς κύματος στην πιο κοντινή

στάθμη πλάτους που έχουμε θέσει και επίσης κάνουμε το ίδιο και για τον χρόνο όπου από συνεχής γίνεται κβαντισμένος, δηλαδή λαμβάνουμε δείγματα ανά μια συγκεκριμένη συχνότητα δειγματοληψίας.



Εικόνα 44: Μετατροπή από Analog σε Digital

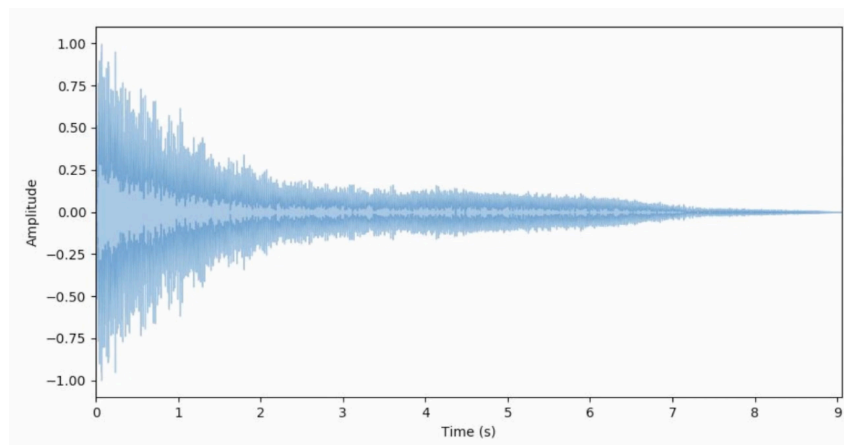
4.2 Μετρικές του Ήχου

Τρεις είναι οι βασικές μετρικές που θα χρησιμοποιήσουμε για τον ήχο:

1. **Sample Rate** ή **Ρυθμός Δειγματοληψίας**. Αναφέρεται στον αριθμό δειγμάτων που λαμβάνονται κάθε δευτερόλεπτο. Σε ένα CD είναι **44,1kHz** αυτό σημαίνει ότι λαμβάνουμε **44.100** δείγματα σε κάθε δευτερόλεπτο.
2. **Bit Depth**. Αυτό αναφέρεται στον αριθμό των bits που αντιστοιχούν σε κάθε δείγμα το οποίο επηρεάζει την δυναμική και το εύρος του ήχου. Ο πιο συχνός αριθμός στην βιομηχανία είναι **16 bits/channel**. Δηλαδή χρησιμοποιούνται 2^{16} στάθμες πλάτους αφού το κάθε πλάτος χαρακτηρίζεται από έναν δυαδικό αριθμό 16 bit.
3. **Bit Rate**. Ο ρυθμός bit είναι ο αριθμός των bit που χρησιμοποιούνται για την αναπαράσταση ενός δευτερολέπτου ήχου. Υπολογίζεται πολλαπλασιάζοντας το Bit Depth με το Sample Rate. Για παράδειγμα, μια ηχογράφηση ήχου **16 bit** σε ρυθμό δειγματοληψίας **44,1kHz** θα έχει ρυθμό bit **705.600 bit** ανά δευτερόλεπτο ($16 \cdot 44.100$).

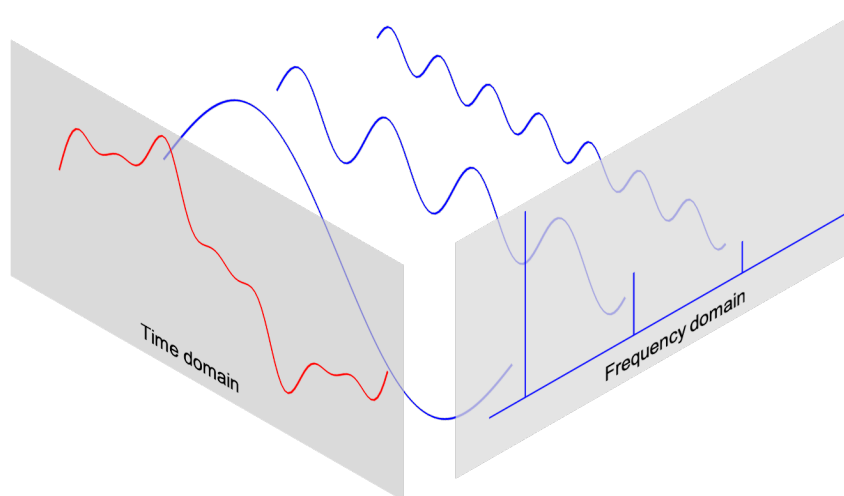
4.3 Fourier Transform

Τώρα θα περάσουμε στο πως πραγματικά είναι ένα κύμα ήχου στην φύση. Όπως φαίνεται στην Εικόνα 45 ένα κύμα ήχου δεν είναι όσο απλό όσο το δείξαμε παραπάνω, στην εικόνα φαίνεται το κύμα που παράγει ένα πλήκτρο πιάνου όπου βλέπουμε ότι ταλαντώνεται με μεγάλο πλάτος στην αρχή και όσο περνάει ο χρόνος αρχίζει και χάνει σε ένταση μέχρις ότου να εξαφανιστεί. Και έτσι προκύπτει το ερώτημα *”Τι μπορούμε να μάθουμε για αυτή την αναπαράσταση κύματος;”*.



Εικόνα 45: Soundwave ενός πλήκτρο από Πιάνο

Την απάντηση στο παραπάνω ερώτημα μας την δίνει το **Fourier Transform**. Αυτό που κάνει ο Μετασχηματισμός Fourier είναι ότι λαμβάνει το παραπάνω κύμα και το μετατρέπει σε ένα άθροισμα ημιτονοειδών κυμάτων όπου το κάθε ένα έχει την δική του συχνότητα, αυτές οι συχνότητες που παράγονται από τον μετασχηματισμό είναι οι συχνότητες που ενυπάρχουν μέσα στο αρχικό κύμα που είδαμε.

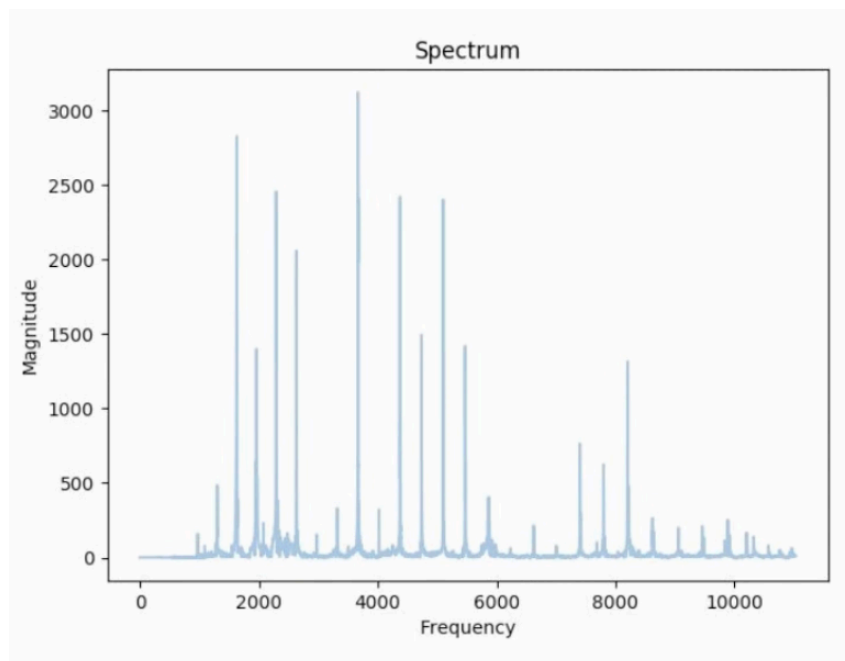


Εικόνα 46: Μετασχηματισμός Fourier

Στην Εικόνα 46 φαίνεται αρχικά το κόκκινο κύμα το οποίο είναι το κύμα εισόδου που εμπεριέχει πολλές συχνότητες, μέσω του μετασχηματισμού Fourier παράγεται μια γραφική παράσταση που φαίνεται δεξιά στο Frequency Domain όπου κάθε γραμμή αναπαριστά και ένα κύμα, το ύψος της γραμμής είναι το πλάτος του κύματος και το που βρίσκεται πάνω στον άξονα frequency έχει να κάνει με την συχνότητα που έχει το κύμα. Αν τα ανάγουμε στον χρόνο τότε φαίνονται τα μπλε κύματα στο κέντρο όπου είναι τα κύματα που αν τα προσθέσουμε θα πάρουμε το αρχικό κόκκινο κύμα. Αυτό που μας ενδιαφέρει κυρίως από τον Μετασχηματισμό είναι το **πλάτος** το εκάστοτε κύματος γιατί αυτό μας δείχνει πόσο το κάθε κύμα συμβάλει στο αρχικό κύμα, ή αλλιώς όσο μεγαλύτερο το πλάτος ενός μεμονωμένου κύ-

ματος τόσο περισσότερο θα επηρεάζει το αρχικό κύμα. Έτσι καταφέρνουμε να γνωρίζουμε ποια συχνότητα είναι η πιο καθοριστική σε ένα σύνθετο κύμα.

Αν εκτελέσουμε Fourier Transform στο κύμα της Εικόνας 45 τότε θα εμφανιστεί το παρακάτω.



Εικόνα 47: Μετασχηματισμός Fourier

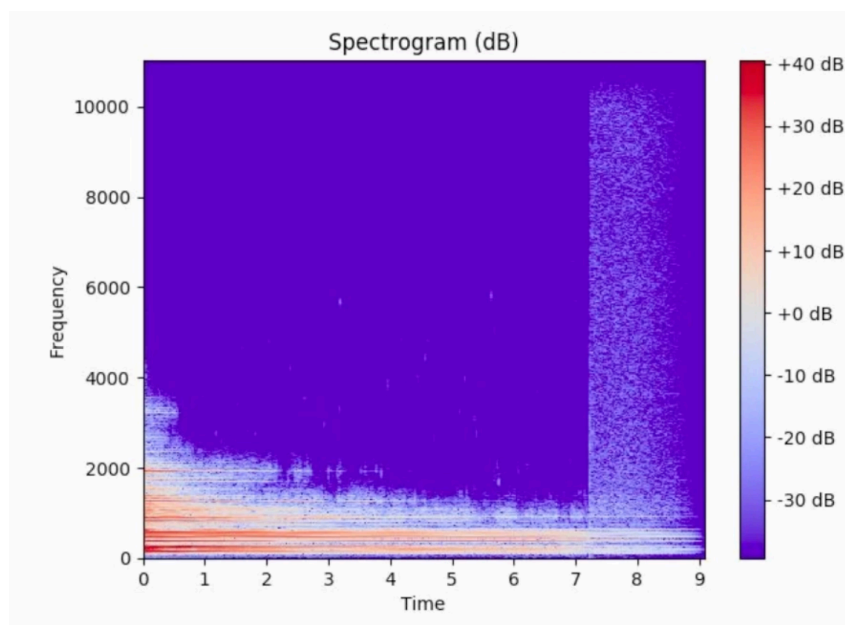
Από την Εικόνα 47 μπορούμε να δούμε ποιες συχνότητες είναι οι πιο καθοριστικές, διακρίνουμε ότι γύρω στα 3.800Hz το πλάτος είναι 3000 το οποίο είναι και το μεγαλύτερο του spectrum. Έτσι καταλαβαίνουμε ότι η συχνότητα που επηρεάζει περισσότερο το κύμα είναι η 3,8kHz.

4.3.1 Short Time Fourier Transform

Ένα αρνητικό με το Fourier Transform ωστόσο είναι ότι δεν λαμβάνουμε γνώση για τον **χρόνο**. Ο Fourier Transform της Εικόνας 47 λαμβάνει υπόψιν όλο το κύμα της Εικόνας 46 για την διάρκεια των 9 δευτερολέπτων των οποίων ταλαντεύεται. Όμως στην μουσική οι συχνότητες, οι δονήσεις και οι ήχοι αλλάζουν με την πάροδο του χρόνου και έτσι παρουσιάζεται η ανάγκη να έχουμε πληροφορία και για τον χρόνο.

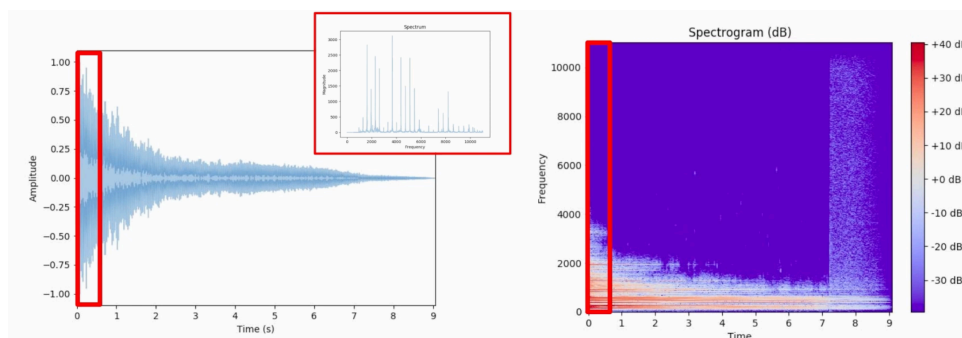
Την λύση σε αυτό το πρόβλημα μας την δίνει το **Short Time Fourier Transform (STFT)**. Αυτό που κάνει το STFT είναι ότι εκτελεί **Fast Fourier Transform (FFT)** ανά χρονικά διαστήματα, αυτό δημιουργεί ένα spectrogram όπου φαίνονται οι συχνότητες, το πλάτος και ο χρόνος. Το αρχικό κύμα διαιρείται σε **frames** όπου σε αυτά τα frames εκτελείται το FFT, στα frames καθορίζουμε το μέγεθος που θέλουμε να έχουν που ονομάζεται **frame size** ή **window size**, όσο μικρότερο το μέγεθος τόσο καλύτερη η ανάλυση του χρόνου αλλά τόσο μειώνεται

και η ανάλυση των συχνοτήτων.



Εικόνα 48: Spectrogram του STFT

Στην εικόνα 48 φαίνεται το Spectrogram του κύματος που παράγεται από το πλήκτρο ενός πιάνο που δείξαμε παραπάνω. Όπως μπορούμε να δούμε πλέον έχουμε άξονα χρόνο όπου φαίνεται στην πάροδο του χρόνου ποιες συχνότητες έχουν τη μεγαλύτερη συμμετοχή, έναν άξονα με συχνότητες και την ένταση της κάθε συχνότητας που φαίνεται με το χρώμα, όσο το χρώμα τείνει στο κόκκινο τόσο μεγαλύτερη και η ένταση της συχνότητας η οποία ένταση μετριέται σε dB. Από αυτό το spectrogram εξάγουμε κάποια συμπεράσματα, ένα ότι πάνω από τα 4000Hz δεν υπάρχει καμία συχνότητα στο κύμα και δεύτερων ότι το εύρος που έχει την μεγαλύτερη συμμετοχή βρίσκεται γύρω στα 0 με 500Hz. Επίσης διακρίνουμε ότι όσο περνάει ο χρόνος η ένταση του spectrogram αρχίζει να φθίνει, όπως έκανε και στο αρχικό κύμα που είδαμε στην Εικόνα 45.



Εικόνα 49: Spectrogram του STFT

Η σειρά που εκτελούνται οι ενέργειες που αναφέραμε παραπάνω φαίνονται στην Εικόνα 49.

1. Πρώτα στο αρχικό κύμα ορίζουμε ένα παράθυρο που φαίνεται με το κόκκινο περίβλημα, αυτό είναι το frame size που ορίσαμε προηγουμένως και αποτελείται από έναν

αριθμό από samples.

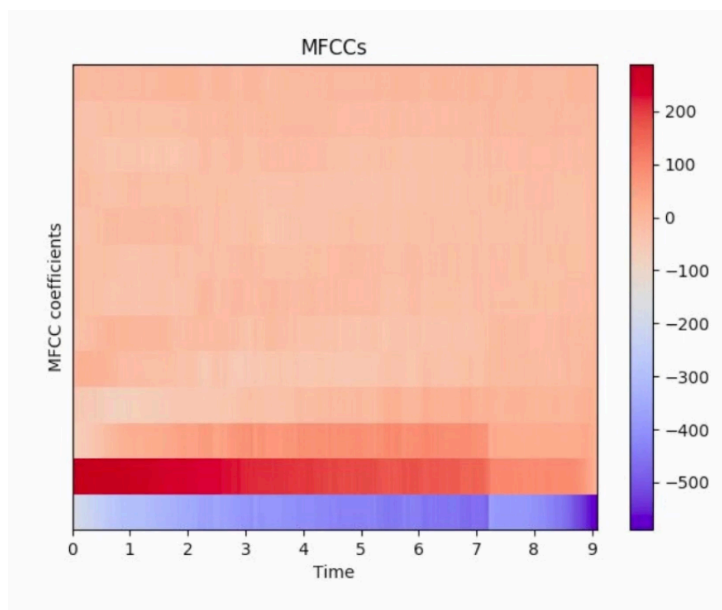
2. Έπειτα σε αυτό το παράθυρο εκτελούμε FFT, φαίνεται στο μικρό παράθυρο στην μέση
3. Τέλος ανάγουμε το FFT στο spectrogram θέτοντας τον χρόνο στον οποίο λάβαμε το δείγμα $[t_0, t_0 + \text{frame_size}]$ για τον άξονα Time, τις συχνότητες του FFT για τον άξονα Frequency και τα πλάτη σε dB και σε χρώμα για την ένταση όπως φαίνεται στο spectrogram.
4. Εκτελούμε τα βήματα 1-3 για όλο το παράθυρο του κύματος κάνοντας shift κάθε φορά ανά frame size.

Το spectrogram που βγαίνει σαν είσοδος από αυτό το pipeline είναι η είσοδος που θα μπει στο Νευρωνικό Δίκτυο μας. Γενικά τέτοιου είδους spectrograms είναι θεμελιώδη μορφές εισόδου για τεχνικές Βαθιάς Μάθησης που έχουν να κάνουν με Ήχο. Όλη η προεπεξεργασία δεδομένων ήχου που σκοπεύουν να χρησιμοποιηθούν από ένα Μοντέλο Βαθιάς Μάθησης περνούν καταλήγουν σε μορφή spectrogram.

4.4 Mel Frequency Cepstral Coefficients - MFCCs

Τα MFCCs είναι μια βασική και θεμελιώδης αναπαράσταση ήχου στις εφαρμογές Βαθιάς Μάθησης που θα μπορούσαμε να πούμε πως είναι πιο σημαντική και από τα Spectrogram. Για να εξάγουμε μια τέτοια πληροφορία από ένα κύμα είναι αρκετά περίπλοκη διαδικασία αλλά αυτό που έχει σημασία να γνωρίζουμε είναι τα MFCC μας δίνουν πληροφορίες που αφορούν το **ηχόχρωμα** και την **υφή** του ήχου. Το ανθρώπινο αυτί δεν αντιλαμβάνεται όλες τις συχνότητες με την ίδια ευαισθησία. Η κλίμακα Mel είναι μια αντιληπτική κλίμακα τόνων που προσεγγίζει την απόκριση του ανθρώπινου αυτιού στις διάφορες συχνότητες. Βασίζεται στη φυσιολογία του αυτιού και αντικατοπτρίζει τον τρόπο με τον οποίο οι άνθρωποι ακούνε τον τόνο. Αν έχουμε για παράδειγμα ένα βιολί και μια τρομπέτα να παίζουν την ίδια μελωδία όπου πιθανών να έχουν το ίδιο Pitch και Συχνότητα αυτό που τα διαφοροποιεί που είναι κάτι που εμείς οι άνθρωποι μπορούμε να το διακρίνουμε είναι η υφή του ήχου ή το ηχόχρωμα, αυτό για να το καταλάβει ένας υπολογιστής χρειάζεται τα MFCC. Αυτό που κάνουν δηλαδή είναι να προσπαθούν να προσομοιώσουν τον μηχανισμό ακοής που έχουμε εμείς οι άνθρωποι.

Για να εκτελέσουμε επεξεργασία MFCC πρώτα περνάμε από ένα μετασχηματισμό Fourier, το οποίο σημαίνει ότι το MFCC είναι πληροφορία στον τομέα της Συχνότητας. Ο τρόπος που αναπαρίστανται τα MFCCs είναι μέσα από έναν αριθμό συντελεστών **coefficients** που συνήθως είναι από 13 έως 40. Αυτά τα coefficient υπολογίζονται σε κάθε frame όπως δείξαμε και πριν στο STFT.



Εικόνα 50: Αναπαράσταση του MFCC

Παραπάνω στην Εικόνα 50 βλέπουμε πως αναπαριστάται το MFCC, είναι παρόμοιο με το spectrogram που είδαμε παραπάνω στο STFT, αφού κάτω φαίνεται ο χρόνος, στα δεξιά φαίνεται η ένταση όπου όσο πιο στο κόκκινο πλησιάζει τόσο μεγαλύτερη ένταση συμβολίζει αλλά τώρα στον άξονα y έχουμε τα MFCC coefficients όπου αυτά αν μετρήσουμε είναι 13, όσα coefficients δηλώσαμε δηλαδή προηγουμένως. Μπορούμε να δούμε ότι έχει παρόμοια φύση με το Spectrogram του STFT καθώς όσο περνάει ο χρόνος η ένταση μειώνεται.

Στην εφαρμογή μας τα MFCCs θα είναι η κύρια δομή δεδομένων που θα χρησιμοποιήσουμε για να εκπαιδεύσουμε το μοντέλο Βαθιάς Μάθησης.

5 Μετρικές για Συστήματα Συστάσεων

Στο κεφάλαιο 2.2 αναφέραμε τις High Level προσεγγίσεις που υπάρχουν στα Content-Based Recommender Systems. Εδώ θα αναφερθούμε στην μέθοδο που επιλέξαμε να υλοποιήσουμε στην εφαρμογή.

Όπως είδαμε στα Content-Based RS βασιζόμαστε στα χαρακτηριστικά του μουσικού κομματιού για να βρίσκουμε τις ομοιότητες. Το πιο συχνό χαρακτηριστικό στο οποίο βασίζονται οι περισσότεροι μελετητές είναι τα MFCC, συνήθως στις εφαρμογές που έχουν την μεγαλύτερη ακρίβεια από τις υπόλοιπες εντάσσονται και συνδυασμοί χαρακτηριστικών. Στην εργασία μας θα χρησιμοποιήσουμε αποκλειστικά τα MFCC για να αναλύσουμε ένα μουσικό κομμάτι και να βρούμε παρόμοια του.

Από ένα κομμάτι για να εξάγουμε τα MFCC πρώτα ορίζουμε τα coefficients, αν π.χ. ορίσουμε 5 coefficients τότε σε κάθε παράθυρο που γίνεται η δειγματοληψία θα λάβουμε έναν πίνακα S από 5 τιμές.

$$S = [-218.3, 138.9, 11.0, 47.1, 31.1]$$

Ένα κομμάτι με βάση το `hop_length` θα διαιρεθεί σε `SAMPLE_RATE * DURATION / hop_length` μέρη, αν π.χ. ένα κομμάτι διαιρείται σε 10 μέρη τότε θα υπάρχει 10 πίνακες S οι οποίοι θα περιέχουν τις τιμές MFCC που θα αναπαριστούν το κομμάτι C .

$$C = (S^1, S^2, \dots, S^{10})$$

Έτσι έχουμε την αναπαράσταση του κομματιού C . Ο τρόπος που θα χρησιμοποιήσουμε για να βρούμε την ομοιότητα μεταξύ κομματιών είναι συγκρίνοντας τα S^{ij} όπου $j \in \{1, 2, \dots, d\}$ ($d = \text{Αριθμός Coefficients}$) μεταξύ των C^i όπου $i \in \{1, 2, \dots, n\}$ ($n = \text{Αριθμός Κομματιών}$). Θα χρησιμοποιήσουμε 2 μεθόδους να συγκρίνουμε τους πίνακες.

5.1 Cosine Similarity

Η ομοιότητα συνημιτόνου μετρά την ομοιότητα μεταξύ δύο διανυσμάτων ενός χώρου εσωτερικού γινομένου. Μετρείται από το συνημίτονο της γωνίας μεταξύ δύο διανυσμάτων και καθορίζει αν δύο διανύσματα έχουν περίπου την ίδια κατεύθυνση. Χρησιμοποιείται συχνά για τη μέτρηση της ομοιότητας εγγράφων στην ανάλυση κειμένου. Θα συγκρίνουμε δύο πίνακες coefficients A και B όπως ορίσαμε παραπάνω. Ο τύπος του cosine similarity φαίνεται παρακάτω στον τύπου 31.

$$|A| |B| \cos \theta = \frac{A \cdot B}{|A| |B|} = \frac{\sum_i^n A_i B_i}{\sqrt{\sum_i^n A_i^2} \sqrt{\sum_i^n B_i^2}} \quad (31)$$

Στην εικόνα 51 φαίνεται το αποτέλεσμα που θα εξαχθεί από την παραπάνω πράξη. Τα διανύσματα A και B έχουν μετατραπεί σε διανύσματα και η γωνία θ μεταξύ τους υποδεικνύει

πόσο παρόμοια είναι.



Εικόνα 51: Γραφικό παράδειγμα cosine similarity

5.2 Euclidean Distance

Ο άλλος τρόπος είναι η Ευκλείδεια Απόσταση. Είναι αρκετά πιο απλή προσέγγιση και ο τύπος της φαίνεται στην εξίσωση 32. Όσο μικρότερος ο αριθμός τόσο μικρότερη και η απόσταση μεταξύ των δύο διανυσμάτων, τόσο πιο παρόμοια και τα MFCC που συγκρίνονται.

$$d(A, B) = \|A - B\| = \sum_{i=1}^d (A_i - B_i)^2 \quad (32)$$

Στην υλοποίηση και οι δύο αλγόριθμοι χρησιμοποιήθηκαν και σε μερικά σημεία εκεί που αποτύγχανε ο ένας πετύχαινε ο άλλος. Κατά μέσο όρο καλύτερη απόδοση είχε ο Cosine Similarity αλγόριθμος αλλά σε μερικά σημεία ο Euclidean Distance είχε καλύτερο accuracy. Ένα θετικό του cosine similarity είναι ότι λαμβάνει υπόψη την κατεύθυνση των διανυσμάτων και όχι το μέγεθός τους. Αυτό μπορεί να είναι επωφελές όταν οι απόλυτες τιμές είναι λιγότερο σημαντικές από τη γωνία μεταξύ των διανυσμάτων. Τα διανύσματα MFCC μπορεί να έχουν διαφορετικά μεγέθη και η ευκλείδεια απόσταση είναι ευαίσθητη σε αυτά τα μεγέθη. Εάν τα μεγέθη δεν είναι κανονικοποιημένα ή εάν οι κλίμακες των χαρακτηριστικών διαφέρουν, αυτό μπορεί να επηρεάσει τις μετρήσεις της απόστασης. Γενικότερα το cosine similarity προτιμάται συχνά σε εργασίες επεξεργασίας ομιλίας και ήχου, όπως η αναγνώριση ομιλίας, επειδή δίνει έμφαση στην ομοιότητα του σχήματος των διανυσμάτων και όχι στα συνολικά επίπεδα ενέργειάς τους.

6 Υλοποίηση & Μεθοδολογία

Σε αυτό το κομμάτι περνάμε στην υλοποίηση της εφαρμογής με βάση όλα τα θεωρητικά θεμέλια που θέσαμε στα προηγούμενα κεφάλαια.

6.1 Εργαλεία

- **Python.** Η εφαρμογή θα υλοποιηθεί με χρήση της γλώσσας Python όπου θα χρησιμοποιηθούν openSource βιβλιοθήκες που θα αναφερθούν στην συνέχεια οι οποίες θα μας προσφέρουν όλα τα απαραίτητα εργαλεία και αλγόριθμους χρειαζόμαστε για να επιτευχθεί ο σκοπός μας.
- **TensorFlow.** Το TensorFlow είναι μια δωρεάν και ανοικτού κώδικα βιβλιοθήκη λογισμικού για μηχανική μάθηση και τεχνητή νοημοσύνη. Μπορεί να χρησιμοποιηθεί σε ένα ευρύ φάσμα εργασιών, αλλά εστιάζει ιδιαίτερα στην εκπαίδευση και την εξαγωγή συμπερασμάτων βαθιών νευρωνικών δικτύων.
- **Keras.** Είναι ένα framework μηχανικής μάθησης το οποίο γράφτηκε σε Python και χρησιμοποιείται σαν Interface της βιβλιοθήκης Tensorflow. Θα το χρησιμοποιήσουμε για να εκπαιδύσουμε το Νευρωνικό Δίκτυο μας. Συνοπτικά, το TensorFlow είναι το υποκείμενο πλαίσιο που παρέχει την υπολογιστική μηχανή και τις λειτουργίες χαμηλού επιπέδου για τη δημιουργία και την εκπαίδευση μοντέλων βαθιάς μάθησης. Το Keras, ενσωματωμένο ως μέρος του TensorFlow, προσφέρει μια φιλική προς το χρήστη, υψηλού επιπέδου διεπαφή για το σχεδιασμό και την κατασκευή πρωτοτύπων νευρωνικών δικτύων. Ο συνδυασμός του TensorFlow και του Keras παρέχει μια ισχυρή και ευέλικτη πλατφόρμα για την ανάπτυξη εφαρμογών μηχανικής μάθησης.
- **PyCharm.** Αυτό θα είναι το IDE στο οποίο θα δουλέψουμε, άλλες εναλλακτικές είναι το Spyder ή το Jupyter Notebook αλλά για την συγκεκριμένη προσέγγιση θα χρησιμοποιήσουμε το PyCharm.
- **Flask.** Το Flask³⁴ θα είναι το κύριο framework το οποίο θα μας χρησιμεύσει για να δημιουργήσουμε την Web εφαρμογή.
- **Librosa.** Το Librosa³⁵ είναι μια βιβλιοθήκη Python για ανάλυση ήχου και μουσικής. Θα είναι η κύρια πηγή αλγορίθμων, κλάσεων και μεθόδων που θα χρησιμοποιούμε για να επεξεργαστούμε το σήμα εισόδου. Μερικές λειτουργίες που μας παρέχει το Librosa είναι:
 - **Audio Loading**, μας δίνει την δυνατότητα να φορτώνουμε αρχεία ήχου διαφόρων format όπως MP3 και WAV, φορτώνει τα αρχεία και τα αποθηκεύει σε NumPy πίνακες όπου στην συνέχεια μπορούν να χρησιμοποιηθούν για ανάλυση.
 - **Feature Extraction**, όπως η εξαγωγή των MFCCs ή χαρακτηριστικά ρυθμού όπως tempo, beat frames κ.α.

³⁴<https://flask.palletsprojects.com/en/3.0.x/>

³⁵<https://github.com/librosa>

- **Spectral Analysis**, μας δίνεται πρόσβαση σε features όπως Short Term Fourier Transform.
- **Visualization**, το librosa μας παρέχει πολλά εργαλεία για να αναπαραστήσουμε τον ήχο και την ανάλυση του όπως τα spectrogram που είδαμε στα προηγούμενα κεφάλαια.
- **Συνεργασία** με βιβλιοθήκες Μηχανικής Μάθησης όπως scikit-learn.
- **Εκτίμηση Pitch** στην πάροδο του χρόνου.

6.2 Προετοιμασία των Δεδομένων

Τα Δεδομένα

Τα δεδομένα τα οποία θα χρησιμοποιήσουμε για την εκπαίδευση του μοντέλου στο Genre Classification είναι το GTZAN Dataset³⁶ το οποίο χρησιμοποιείται για αυτόν τον σκοπό και είναι από τα πιο δημοφιλή στον χώρο του MGC. Το Dataset αποτελείται από τα παρακάτω:

- 10 Είδη μουσικής (*Pop, Rock, Disco, ...*) όπου το κάθε ένα περιέχει 100 τραγούδια όπου διαρκεί 30 δευτερόλεπτα έκαστο. Όλα είναι σε μορφή .wav.

> rock	
> reggae	
> pop	
> metal	rock.00026.wav 1,3 MB
> jazz	rock.00025.wav 1,3 MB
> hiphop	rock.00024.wav 1,3 MB
> disco	rock.00023.wav 1,3 MB
> country	rock.00022.wav 1,3 MB
> classical	rock.00021.wav 1,3 MB
> blues	rock.00020.wav 1,3 MB
	rock.00019.wav 1,3 MB
	rock.00018.wav 1,3 MB
	rock.00017.wav 1,3 MB
	rock.00016.wav 1,3 MB

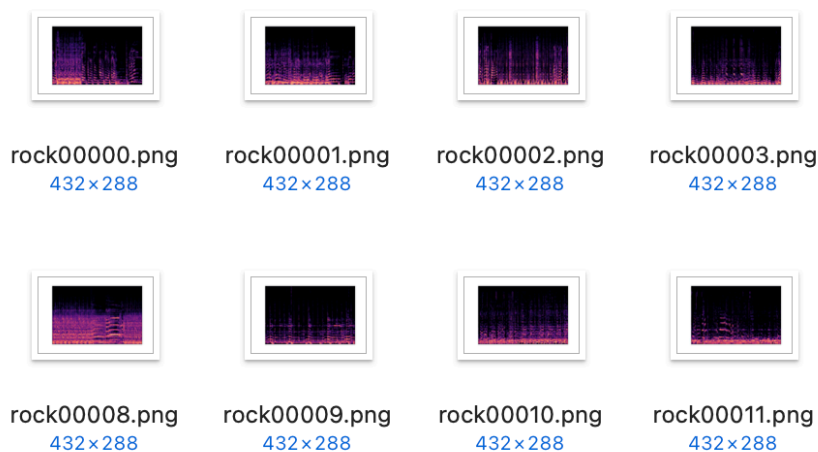
(i) Είδη Μουσικής

(ii) Κομμάτια που περιέχονται

Εικόνα 52: Κατάλογος τραγουδιών.

- Ένα φάκελο με οπτική αναπαράσταση για κάθε αρχείο ήχου. Ένας τρόπος ταξινόμησης των δεδομένων είναι τα νευρωνικά δίκτυα. Επειδή τα NN συνήθως λαμβάνουν κάποιο είδος αναπαράστασης εικόνας, τα αρχεία ήχου μετατράπηκαν σε Mel Spectrograms για να γίνει αυτό εφικτό. Εμείς στην εφαρμογή μας όμως θα δημιουργήσουμε τις δικές μας αναπαραστάσεις εικόνας.

³⁶<https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification>



Εικόνα 53: Εικόνες των Κομματιών

- 2 Αρχεία CSV τα οποία περιέχουν Features των αρχείων ήχου. Εμπεριέχουν για κάθε τραγούδι (διάρκειας 30 δευτερολέπτων) μια μέση τιμή και μια διακύμανση που υπολογίζονται σε πολλαπλά χαρακτηριστικά που μπορούν να εξαχθούν από ένα αρχείο ήχου.

Εμείς από όλα τα παραπάνω θα χρησιμοποιήσουμε απλά τα αρχεία ήχου και θα τα αναλύσουμε και θα τα μετατρέψουμε στην μορφή που επιθυμούμε χωρίς να ασχοληθούμε με τα άλλα που περιέχονται στο Dataset.

Εξαγωγή χαρακτηριστικών

Στον κώδικα που φαίνεται στο παράρτημα 5 γίνεται η προεπεξεργασία των δεδομένων. Ο τρόπος που θα επιτελέσουμε αυτόν τον σκοπό είναι εξάγωντας τα MFCC του κάθε κομματιού και αποθηκεύοντας τα σε ένα JSON αρχείο για να είναι μονίμως διαθέσιμο. Τα MFCC εξάγονται μέσω της βιβλιοθήκης **librosa** η οποία μας προσφέρει έτοιμες μεθόδους όχι μόνο για να πάρουμε την MFCC αναπαράσταση από ένα ηχητικό σήμα, αλλά και STFT, Chromagram και Rhythm και άλλα. Σε κάθε εγγραφή διανύσματος MFCC θα είναι αποθηκευμένο και το label που σηματοδοτεί σε ποιο είδος ανήκει αυτό το MFCC. Σε αυτό το σημείο θα ορίσουμε τις ρυθμίσεις που επιθυμούμε όπως φαίνονται στον Πίνακα 1, αυτές οι τιμές εισάγονται στην μέθοδο του librosa για να μας παράξει τα MFCC.

n_mfcc	Ο αριθμός coefficient που θα χρησιμοποιηθούν στο MFCC spectrogram, εμείς θα χρησιμοποιήσουμε 13.
n_fft	Είναι το παράθυρο στο οποίο πραγματοποιείται ο FFT όπως είδαμε στο ανάλογο κεφάλαιο προηγουμένως. Εδώ θέτουμε το παράθυρο να περιέχει 2048 samples.
hop_length	Το βήμα κατά το οποίο μετακινείται το παράθυρο, σε κάθε επανάληψη που εκτελείται ο FFT μέχρι να φτάσει στο τέλος του δείγματος ήχου το παράθυρο μετακινείται κατά 512 samples.

sample_rate Τα δείγματα που λαμβάνονται το δευτερόλεπτο, στην περίπτωση μας 22050.

Πίνακας 1: Παράμετροι του preprocess

Λόγω μικρού μεγέθους δεδομένων χρειάζεται να αυξήσουμε εικονικά την βάση. Για αυτό τον σκοπό χωρίζουμε το κάθε κομμάτι σε segments όπου στην υλοποίηση τον αριθμό των segments τον ορίσουμε σε 10. Αυτό σημαίνει ότι θα επαναληφθεί η πλήρης διαδικασία εξαγωγής MFCC **δέκα φορές** ανά κομμάτι, αυτό μπορούμε να το φανταστούμε σαν ένα παράθυρο όπου κάθε φορά μετακινείται κατά ένα ποσό προς τα μπροστά και σε κάθε επανάληψη εξάγει MFCC από το κομμάτι και τους φέρεται σαν να είναι ολότελα ξεχωριστό τραγούδι από εκεί που ήταν πριν το window. Χρειάζεται επειδή δεν είναι όλα τα κομμάτια ίδιου μήκους να ελέγξουμε και για το μήκος έτσι ώστε να έχουμε ομοιογένεια στα δεδομένα εισόδου.

Παρακάτω βλέπουμε σε ποια μορφή βρίσκονται τα MFCC.

```
{
  "mapping": [
    "pop",
    "metal",
    ...
    "jazz"
  ],
  "mfcc": [
    [
      [
        -218.3363037109375,
        138.9757080078125,
        11.036369323730469,
        47.1761474609375,
        31.118013381958008,
        7.663910865783691,
        -14.681863784790039,
        -25.850101470947266,
        16.493316650390625,
        -5.018362998962402,
        -16.807178497314453,
        -5.647577285766602,
        -24.711021423339844
      ],
      [
        -200.65347290039062,
        135.44236755371094,
        5.9279069900512695,
        51.504798889160156,
        32.537147521972656,
        1.3401339054107666,
        -16.875591278076172,
        -31.766538619995117,
        6.489632606506348,
        -10.715614318847656,
        -13.535757064819336,
        -8.524260520935059,
        -18.94091033935547
      ],
      ...
    ]
  ],
  "labels": [
    0,
    0,
    0,
    ...
    9,
    9
  ]
}
```

Όπως μπορούμε να παρατηρήσουμε στην κατηγορία `mapping` έχουν αποθηκευτεί τα στυλ μουσικής. Παρακάτω στα `mfcc` βρίσκονται σε μορφή διανύσματος 13 τιμές, ο αριθμός 13 προκύπτει από τον αριθμό `n_mfcc` που ορίσαμε παραπάνω στο πρόγραμμα. Το κάθε `mfcc[i]` αντιστοιχεί και σε ένα `labels[i]` όπου το `labels[i]` προσδιορίζει σε ποιο είδος ανήκει το `mfcc[i]`.

Το παραπάνω JSON αρχείο τώρα θα το περάσουμε από το πρόγραμμα όπου θα δημιουργήσουμε ένα νευρωνικό δίκτυο και θα το εκπαιδεύσουμε με τα δεδομένα που εξάγαμε παραπάνω.

Χωρισμός των Δεδομένων

Για την εκπαίδευση των μοντέλων που ακολουθεί εκτός από τα τυπικά Training και Test Sets θα χρησιμοποιήσουμε και Validation Set. Το Validation Set είναι ένας ενδιάμεσος μεταξύ του Training Set και Test Set. Σκοπός του είναι η λεπτομερής ρύθμιση των hyperparameters του μοντέλου και η αξιολόγηση της απόδοσής του κατά τη διάρκεια της εκπαίδευσης. Σε αντίθεση με το Test Set, το Validation Set χρησιμοποιείται επαναληπτικά κατά τη διάρκεια της ανάπτυξης του μοντέλου. Αξιολογώντας τις επιδόσεις του μοντέλου στο Validation Set, μπορούμε να λάβουμε τεκμηριωμένες αποφάσεις σχετικά με την προσαρμογή των hyperparameters, την επιλογή του μοντέλου με τις καλύτερες επιδόσεις ή τον εντοπισμό πιθανών προβλημάτων overfitting ή underfitting. Έτσι θα χωρίσουμε τα δεδομένα σε Training, Validation και Test Data. Πιο συγκεκριμένα το **Training Set** θα πάρει το **55%** του Data, το **Validation Set** το **25%** των Data και τέλος το **Test Set** θα πάρει το **20%** των Data.

Στην εκπαίδευση των μοντέλων θα εφαρμοστούν **50 epochs**, και θα χρησιμοποιήσουμε την **Mini-Batch** προσέγγιση με αριθμό από Batch **32**.

6.3 Μετρικές

Για να αξιολογήσουμε τα μοντέλα που θα δημιουργήσουμε θα χρειαστούμε και τις ανάλογες μετρικές. Οι μετρικές που θα χρησιμοποιήσουμε θα είναι οι: **Accuracy**, **Precision**, **Recall** και **F1-Score**.

- **Accuracy**: Αυτή η μετρική μετρά το ποσοστό των σωστών προβλέψεων που έκανε το μοντέλο σε όλο το σύνολο των δεδομένων. Απαντάει στο ερώτημα "Από όλες τις προβλέψεις που κάναμε, πόσες ήταν αληθινές;"

$$accuracy = \frac{true\ positives + true\ negatives}{true\ positives + true\ negatives + false\ positives + false\ negatives}$$

- **Precision**: Μετράει το ποσοστό των True Positive προβλέψεων μεταξύ όλων των Positive προβλέψεων του μοντέλου. Απαντάει στο "Από όλες τις θετικές προβλέψεις που κάναμε, πόσες επαληθεύτηκαν;"

$$precision = \frac{true\ positives}{true\ positives + false\ positives}$$

- **Recall:** Επικεντρώνεται στο πόσο καλό είναι το μοντέλο στην εύρεση όλων των θετικών αποτελεσμάτων. Απαντά στην ερώτηση "Από όλα τα σημεία δεδομένων που θα έπρεπε να προβλεφθούν ως αληθή, πόσα προβλέψαμε σωστά ως αληθή;".

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

- **F1-Score:** Είναι μια μετρική που συνδυάζει την ανάκληση και την ακρίβεια. Όπως είδαμε, υπάρχει ένας συμβιβασμός μεταξύ ακρίβειας και ανάκλησης, το F1 μπορεί επομένως να χρησιμοποιηθεί για να μετρήσει πόσο αποτελεσματικά τα μοντέλα μας κάνουν αυτόν τον συμβιβασμό.

$$F1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

6.4 Μοντέλα Βαθιάς Μάθησης

6.4.1 Multilayer Perceptron Network

Σε αυτό το σημείο θα δημιουργήσουμε το Multilayer Perceptron μοντέλο. Για να ορίσουμε την αρχιτεκτονική χρησιμοποιούμε την βιβλιοθήκη **keras** η οποία μας προσφέρει έναν εύκολο τρόπο για να δημιουργήσουμε ένα νευρωνικό δίκτυο. Το μοντέλο του **keras** που θα χρησιμοποιηθεί είναι το **keras.Sequential**³⁷ το οποίο δημιουργεί ένα νευρωνικό δίκτυο με διαδοχική λογική δηλαδή οι νευρώνες του Hidden Layer i δέχονται input από το Layer $i - 1$ και δίνουν έξοδο στους νευρώνες του $i + 1$, με λίγα λόγια η κίνηση στο νευρωνικό δίκτυο γίνεται γραμμικά από τα αριστερά προς τα δεξιά. Επίσης ορίσουμε να ισχύει **Dense** σύνδεση δηλαδή κάθε νευρώνας του Hidden Layer i να συνδέεται με κάθε νευρώνα του Hidden Layer $i + 1$.

Input Layer

Για το Input Layer θα χρειαστεί να γνωρίζουμε από πόσους νευρώνες θα αποτελείται. Όπως είπαμε και παραπάνω το `SAMPLE_RATE = 22050`, η διάρκεια των κομματιών είναι 30 sec άρα κάθε κομμάτι αποτελείται από $22050 \cdot 30 = 661500$ samples. Εφόσον θα δουλέψουμε με segments και κάθε κομμάτι σπάει σε 10 segment το κάθε segment ως αποτέλεσμα θα περιέχει $661500 / 10 = 66150$ samples. Από αυτά τα samples θα εξαχθούν τα MFCC. Όπως είπαμε έχουμε τα MFCC λαμβάνονται σε ένα παράθυρο του κομματιού και όχι σε όλο το κομμάτι, αυτό το παράθυρο έχει μήκος προχωράει κατά `hop_length` όπου στην υλοποίηση μας `hop_length = 512`. Έτσι για ένα segment θα υπάρχουν συνολικά $\lceil 66150 / 512 \rceil = 130$ διανύσματα από 13 MFCC. Έτσι τα δεδομένα εισόδου είναι σε μορφή (130, 13). Αυτά θα τα κάνουμε **flatten** δηλαδή θα τα μετατρέψουμε σε διάνυσμα, το τελικό διάνυσμα θα έχει μέγεθος $130 \cdot 13 = 1690$, άρα το Input Layer θα αποτελείται από 1690 νευρώνες.

Hidden Layers

Για τα Κρυφά Στρώματα θα χρησιμοποιήσουμε πάλι Dense αρχιτεκτονική. Στην συνέχεια θα ορίσουμε τον αριθμό των νευρώνων που θέλουμε κάθε layer να έχει. 512 στο πρώτο layer,

³⁷<https://keras.io/api/models/sequential/>

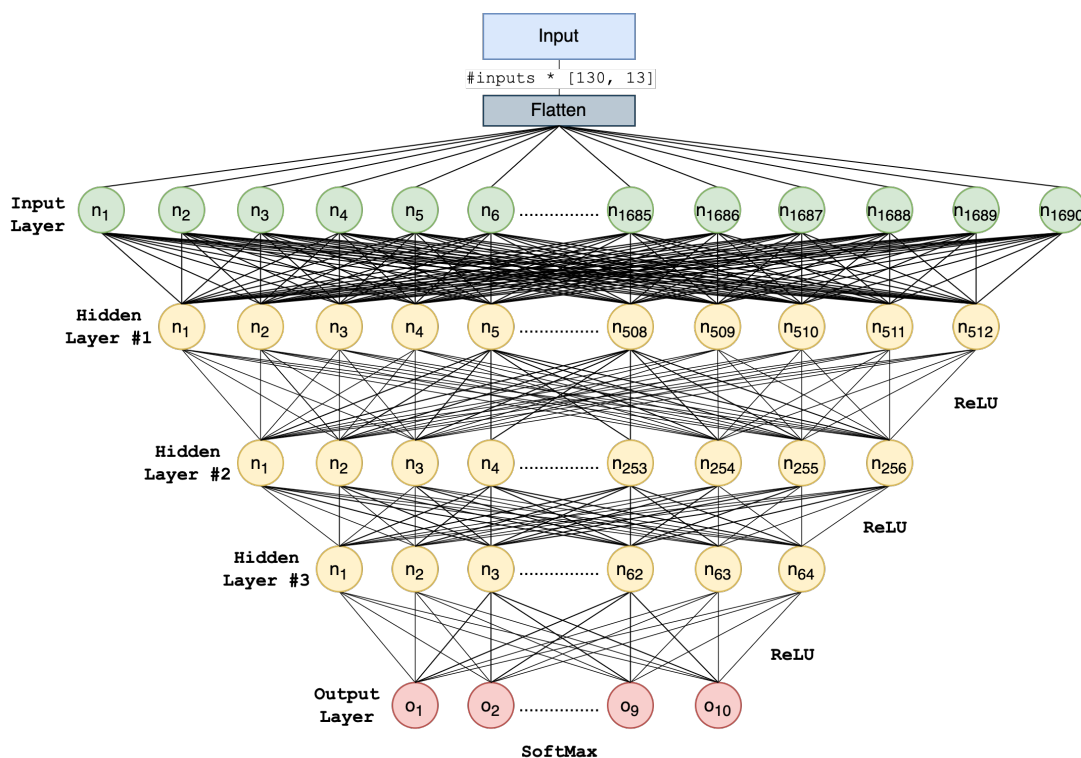
256 στο δεύτερο και 64 στο τρίτο. Ορίζουμε επίσης και την συνάρτηση ενεργοποίησης που επιθυμούμε που σε αυτή την περίπτωση είναι η ReLU, της οποίας τα θετικά και ο λόγος που την διαλέγουμε βρίσκεται στο ανάλογο κεφάλαιο.

Output Layer

Τέλος το layer εξόδου θα αποτελείται από 10 Νευρώνες, ο αριθμός προκύπτει από τα είδη μουσικής που αναμένουμε να έχουμε σαν συμπέρασμα του Νευρωνικού Δικτύου, αν είχαμε 11 είδη μουσικής θα είχαμε 11 νευρώνες εξόδου. Εδώ θα χρησιμοποιήσουμε την **softmax** συνάρτηση ενεργοποίησης αφού έχουμε ένα πρόβλημα του στυλ classification και θέλουμε οι δέκα νευρώνες στην έξοδο να λάβουν ο καθένας από ένα ποσοστό το οποίο αφορά κατά πόσο τα δεδομένα που εισήχθησαν ανήκουν στο κάθε είδος δηλαδή κάθε νευρώνα.

Compile και Train του Network

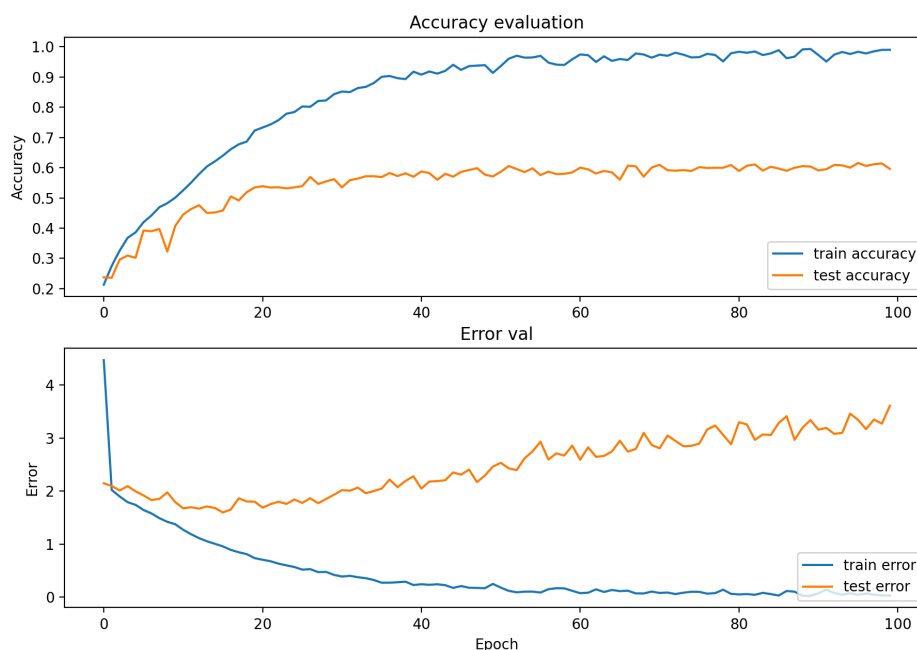
Όταν καλούμε να κάνουμε compile το δίκτυο, πρώτα δημιουργούμε έναν **optimiser** όπου εδώ διαλέγουμε τον Adam. Ο optimiser διασφαλίζει ότι χρησιμοποιούνται κατάλληλα βάρη και συναρτήσεις απώλειας ώστε να διατηρείται βελτιστοποιημένη η διαφορά μεταξύ της προβλεπόμενης και της πραγματικής τιμής του μοντέλου μάθησης του νευρωνικού δικτύου. Ο Adam είναι μια επέκταση του Stochastic Gradient Descent και σύμφωνα με τους Kingma *et al.*, 2014, η μέθοδος είναι "υπολογιστικά αποδοτική, έχει μικρή απαίτηση μνήμης, παραμένει αναλλοίωτη στη διαγώνια αναπροσαρμογή των gradients και είναι κατάλληλη για προβλήματα που είναι μεγάλα από άποψη δεδομένων/παραμέτρων". Για error function ορίζουμε την "sparse_categorical_crossentropy".



Εικόνα 54: Αρχιτεκτονική του MultiLayer Perceptron Network

Επίλυση του Overfitting

Η πρόοδος του μοντέλου στο πέρας των epochs φαίνεται στην Εικόνα 55.



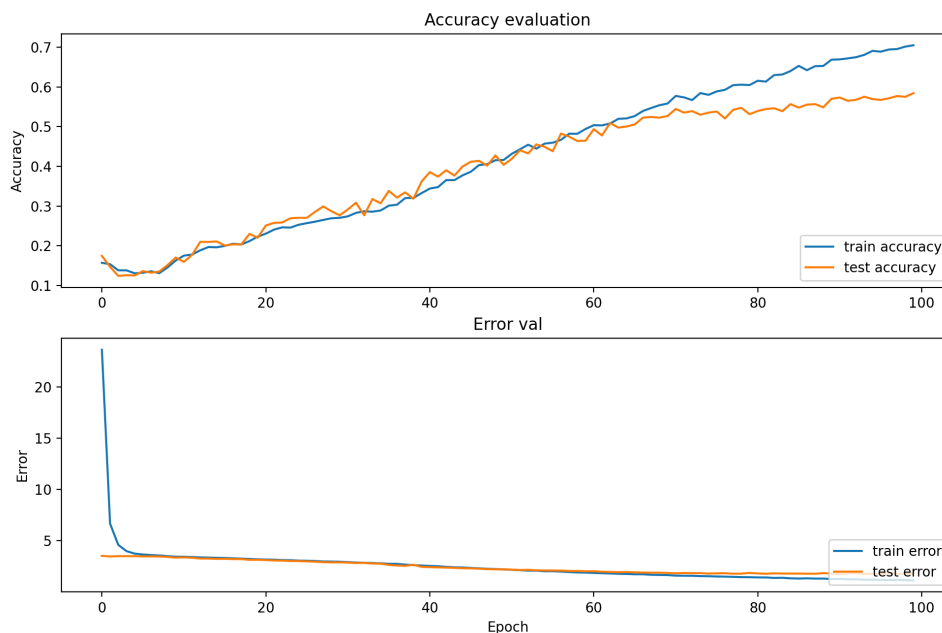
Εικόνα 55: Εκπαίδευση του μοντέλο ανά Epoch

Παρουσιάζεται το πρόβλημα του Overfitting που αναφέραμε στο Κεφάλαιο 3.6, βλέπουμε ότι όσο προχωράνε τα Epochs το Accuracy του train data τείνει να παραμένει σταθερό ενώ το Error αυξάνεται, και αυτό όσο η ακρίβεια στα Training Data αυξάνεται κατά πολύ. Για να επιλύσουμε αυτό το πρόβλημα θα χρησιμοποιήσουμε τις μεθόδους **Dropout** και **L2 Regularization**. Για να επιτευχθούν αυτές οι εντολές με την χρήση του Tensorflow είναι πολύ απλό και φαίνεται στον Κώδικα 5. Η τιμή 0.001 που ορίζουμε στον regularizer είναι το λ που αναφέραμε παραπάνω. Στο `keras.layers.Dropout(p)` ορίζουμε σαν p την πιθανότητα απενεργοποίησης που θέλουμε ο κάθε νευρώνας του ανάλογου Layer να έχει στην περίπτωση μας το ορίζουμε σε 30%. Τώρα η νέα γραφική παράσταση είναι η Εικόνα 62.

6.4.2 Convolutional Neural Network

Θα μετατρέψουμε το δίκτυο μας από Multilayer Perceptron σε Convolutional Neural Network. Ο κώδικας είναι ο 8. Ο αλγόριθμος του CNN όπως αναφέραμε αναμένει δεδομένα που βρίσκονται σε μορφή 3D, εδώ τα δεδομένα μας είναι σε δισδιάστατη μορφή (130, 13) όπου 130 είναι τα time bins από το κάθε κομμάτι και το 13 είναι ο αριθμός MFCC. Εμείς αυτό που πρέπει να κάνουμε είναι να προσθέσουμε μια ακόμη διάσταση, να το μετατρέψουμε δηλαδή σε (130, 13, 1). Αυτό που δημιουργείται είναι μια 4D μεταβλητή της μορφής (`sum_samples, 130, 13, 1`)

Στην υλοποίηση μας το μοντέλο θα αποτελείται από 3 Convolutional Layers όπου το κάθε ένα θα ακολουθείται από 1 Max Pooling Layer. Στην δημιουργία του Convolutional Layer ορίζουμε:



Εικόνα 56: Error & Accuracy μετά από Regularization

1. Πόσα Kernels/Filters θα έχει, εμείς εδώ ορίζουμε 32.
2. Διαστάσεις του Kernel, εμείς θέτουμε 3x3
3. Την συνάρτηση ενεργοποίησης, ορίζουμε την ReLU
4. Το σχήμα που θα έχει το Layer όπου στην περίπτωση μας είναι τρισδιάστατο και το βρίσκουμε χρησιμοποιώντας την μέθοδο `.shape()` στην `inputs_train` μεταβλητή. Αυτό που επιστρέφεται πρακτικά είναι:
 - `inputs_train.shape[0]` = 5492, Αριθμός συνολικών εγγραφών για εκπαίδευση του μοντέλου.
 - `inputs_train.shape[1]` = 130, Time Bins.
 - `inputs_train.shape[2]` = 13, Αριθμός MFCC.
 - `inputs_train.shape[3]` = 1, επιπλέον διάσταση που απαιτείται από το CNN.

Στην συνέχεια θα προσθέσουμε ένα Layer Max Pooling. Εκεί θα θέσουμε σαν ρυθμίσεις το Pool Size, Strides με άλμα 2x2, Padding same δηλαδή να προστεθούν τιμές από όλες τις πλευρές. Τελευταίο Layer θα προσθέσουμε το Batch Normalization το οποίο έχει ένα αρκετά σύνθετο μαθηματικό υπόβαθρο αλλά αυτό που μας ενδιαφέρει στην προκειμένη περίπτωση είναι ότι επιταχύνει την σύγκλιση των βαρών του δικτύου και ως αποτέλεσμα βελτιώνει κατά πολύ την ταχύτητα της εκπαίδευσης. Τις ρυθμίσεις του πρώτου Layer θα τις αντιγράψουμε και για το δεύτερο Layer, στο τρίτο Layer θα αλλάξουμε απλά τις διαστάσεις του Kernel του Convolutional Layer και του Max Pooling.

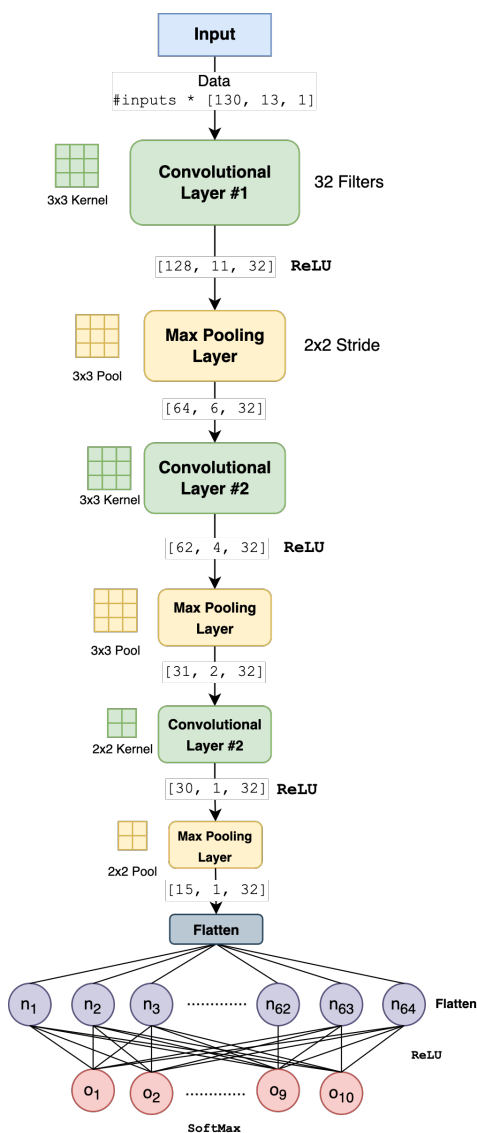
Πριν το Output Layer μετατρέπουμε τον πίνακα σε διάνυσμα πολύ απλά μέσω της Flatten μεθόδου του keras. Σε αυτό το σημείο είναι που μπαίνει το πρώτο Dense Layer που είναι

πλήρως συνδεδεμένο με το Flatten Layer. Για τα Dense Layer θα ακολουθήσουμε παρόμοια υλοποίηση όπως κάναμε και στα Υποκεφάλαια 6.4.1 και 6.4.1, θα ορίσουμε δηλαδή αριθμό νευρώνων, συνάρτηση ενεργοποίησης και πιθανότητα Dropout για αποφυγή του Overfitting. Στο Dense Layer του CNN θα βάλουμε 64 νευρώνες με συνάρτηση ενεργοποίησης ReLU.

Τέλος για το layer εξόδου θα βάλουμε 10 νευρώνες όσα και τα είδη μουσικής με συνάρτηση ενεργοποίησης softmax που αφορά multiclass classification προβλήματα.

Παρακάτω φαίνεται η δομή του CNN

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 128, 11, 32)	320
max_pooling2d (MaxPooling2D)	(None, 64, 6, 32)	0
batch_normalization (Batch Normalization)	(None, 64, 6, 32)	128
conv2d_1 (Conv2D)	(None, 62, 4, 32)	9248
max_pooling2d_1 (MaxPooling2D)	(None, 31, 2, 32)	0
batch_normalization_1 (Batch Normalization)	(None, 31, 2, 32)	128
conv2d_2 (Conv2D)	(None, 30, 1, 32)	4128
max_pooling2d_2 (MaxPooling2D)	(None, 15, 1, 32)	0
batch_normalization_2 (Batch Normalization)	(None, 15, 1, 32)	128
flatten (Flatten)	(None, 480)	0
dense (Dense)	(None, 64)	30784
dropout (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 10)	650

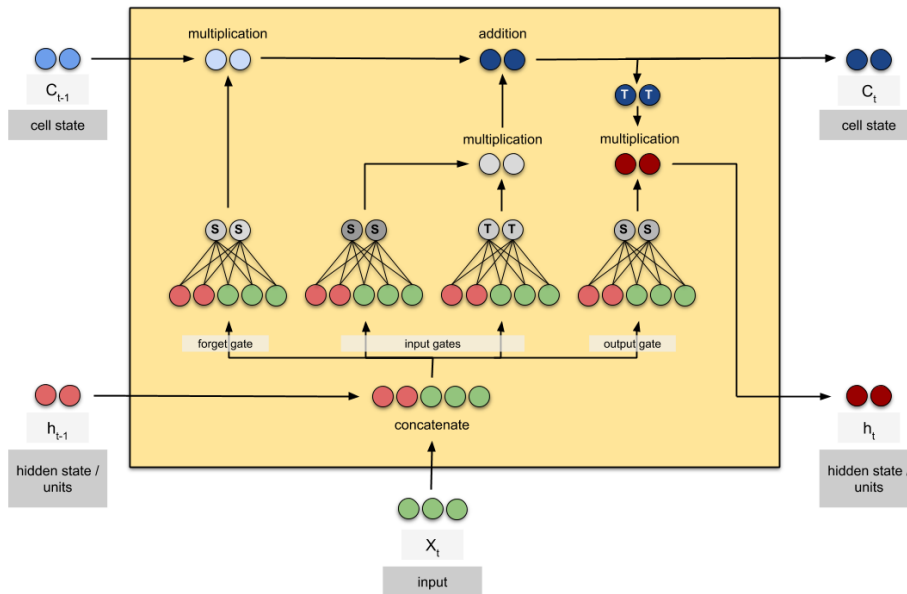


Εικόνα 57: Η αρχιτεκτονική του CNN μοντέλου

6.4.3 Recurrent Neural Network - LSTM

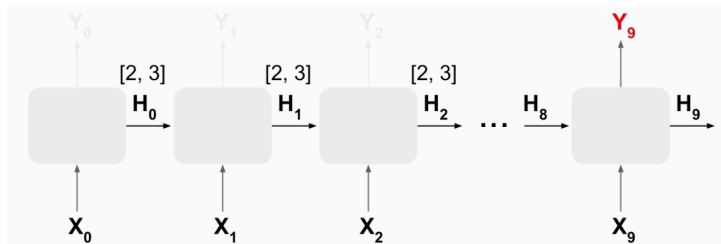
Τελικό μοντέλο που θα υλοποιήσουμε θα είναι το Long Short Term Memory, LSTM Neural Network. Πάλι για να το δημιουργήσουμε θα χρησιμοποιήσουμε την βιβλιοθήκη keras του Tensorflow. Η διαφορά στην υλοποίηση σε σχέση με το CNN είναι ότι δεν χρειαζόμαστε μια τρίτη διάσταση στα δεδομένα εισόδου που στο CNN αφορούσαν το βάθος των δεδομένων.

Για το LSTM νευρωνικό δίκτυο θα χρησιμοποιήσουμε 2 LSTM επίπεδα. Το πρώτο επίπεδο θα έχει **64 units**. Τα units φαίνονται στην Εικόνα 58, τα units είναι ο αριθμός των νευρώνων του hidden state, αυτό που εισάγεται δηλαδή στον πυρήνα και αυτό που εξάγεται από τον πυρήνα μετά από επεξεργασίες, είναι λογικό να συμπεράνουμε ότι όσο μεγαλύτερος ο αριθμός των units τόση περισσότερη πληροφορία θα μπορεί να συλλέγει το hidden state του LSTM. Στην δική μας την περίπτωση 64 units σημαίνει ότι το επίπεδο LSTM έχει 64 κύτταρα μνήμης ή νευρώνες.

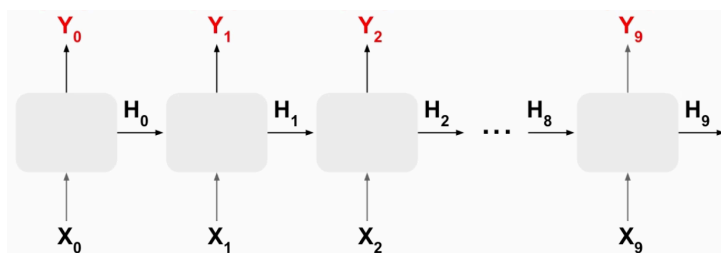


Εικόνα 58: Units σε ένα LSTM
<https://towardsdatascience.com/>

Κάτι ακόμα που εισάγουμε είναι την λογική του `return_sequence`. Η `request_sequence` παράμετρος παίζει ρόλο στον καθορισμό του τι θα πρέπει να εξάγει το επίπεδο LSTM. Αν την θέσουμε `False` τότε το επίπεδο LSTM θα επιστρέψει μόνο την έξοδο για το τελευταίο χρονικό βήμα. Αυτό σημαίνει ότι το LSTM επεξεργάζεται ολόκληρη την ακολουθία εισόδου και παρέχει μια περίληψη της κατανόησής του στο τελικό χρονικό βήμα. Αν την θέσουμε σε `True` τότε το LSTM θα εξάγει τα hidden layers για κάθε χρονικό βήμα στην ακολουθία. Αυτό μπορεί να είναι χρήσιμο όταν στοιβάζονται πολλαπλά στρώματα LSTM και θέλουμε κάθε στρώμα να έχει πρόσβαση σε ολόκληρη την ακολουθία των κρυφών καταστάσεων.



Εικόνα 59: Sequence to Vector LSTM



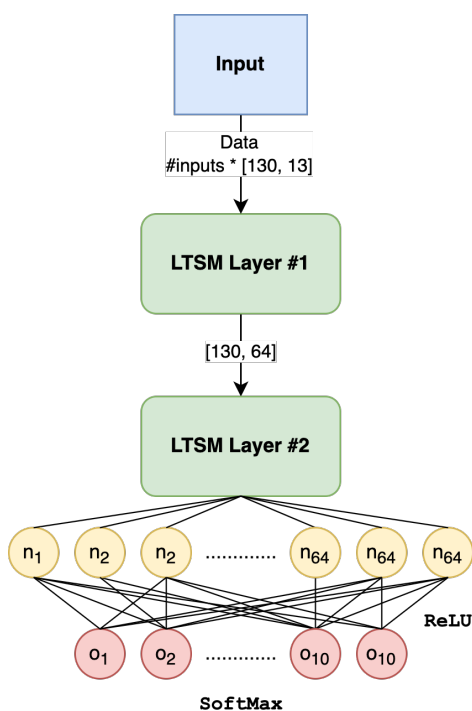
Εικόνα 60: Sequence to Sequence LSTM

Ο λόγος που επιλέγουμε Sequence to Sequence αρχιτεκτονική είναι γιατί θέλουμε τα Hidden States να περαστούν στο **δεύτερο** στρώμα LSTM. Έτσι θα συνδέσουμε το πρώτο στρώμα LSTM δημιουργήσουμε σε ένα δεύτερο στρώμα LSTM όπου και αυτό θα έχει 64 units.

Μετά από το δεύτερο στρώμα θα εισάγουμε ένα Dense Layer το οποίο θα έχει 64 νευρώνες και θα χρησιμοποιεί ReLU συνάρτηση ενεργοποίησης. Θα εφαρμόσουμε Dropout στο Dense Layer έτσι ώστε να αποφευχθεί η υπερπροσαρμογή. Το summary του μοντέλου είναι το παρακάτω και φαίνεται και στην Εικόνα 61.

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 130, 64)	19968
lstm_1 (LSTM)	(None, 64)	33024
dense (Dense)	(None, 64)	4160
dropout (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 10)	650

=====
 Total params: 57802 (225.79 KB)
 Trainable params: 57802 (225.79 KB)
 Non-trainable params: 0 (0.00 Byte)



Εικόνα 61: Τελικό LSTM μοντέλο

6.5 Μηχανή δημιουργίας Συστάσεων

6.5.1 Με χρήση MFCC

Για να χειριστούμε τα αρχεία ήχου θα τα αναπαραστήσουμε σε μορφή MFCC όπως κάναμε και στο MGC, μόνο που τώρα δεν θα χρησιμοποιήσουμε segments, δηλαδή τα MFCC θα εξαχθούν **μια φορά ανά κομμάτι**. Επίσης θα αυξήσουμε τα MFCC coefficients σε **40** ώστε να υπάρχει μεγαλύτερη ακρίβεια για τα συστατικά του ήχου από τα οποία αποτελείται το κομμάτι. Οι πληροφορίες θα αποθηκευτούν πάλι σε JSON αρχείο που τώρα θα αποθηκευτούν και τα path των κομματιών. Η φιλοσοφία της προσέγγισης θα είναι ότι ο χρήστης θα εισάγει ένα μουσικό κομμάτι και θα του προτείνει παρόμοια μουσική από τα 1000 τραγούδια που έχουμε στην διάθεση μας. Ο κώδικας του MFCC extractor είναι ο 6.

Η μορφή του JSON είναι η παρακάτω, κάθε mfcc εγγραφή αποτελείται από 40 νούμερα το οποίο οφείλεται στο ότι ορίσαμε 40 coefficients.

```
{
  "name": [
    "pop.00027.wav",
    "pop.00033.wav",
    ...
    "metal.00022.wav",
  ],
  "mfcc": [
    [
      [
        -218.3363037109375,
        138.9757080078125,
        ...
      ],
      ...
      [
        -230.8790283203125,
        130.29129028320312,
        ...
      ],
    ],
  ],
  "path": [
    "/Users/conspd/School/Thesis/Data/genres_original/pop",
    "/Users/conspd/School/Thesis/Data/genres_original/pop",
    ...
    "/Users/conspd/School/Thesis/Data/genres_original/jazz"
  ]
}
```

Δημιουργία του Recommendation Engine

Έχουμε την βάση με τα MFCC όλων των κομματιών που διαθέτουμε και τώρα επιθυμούμε ο χρήστης να είναι σε θέση να εισάγει ένα μουσικό κομμάτι και να του γίνονται οι ανάλογες συστάσεις. Από το κομμάτι που θα εισάγει ο χρήστης θα εξάγουμε ένα παράθυρο 30 δευτερολέπτων το οποίο θα συγκρίνεται με τα υπόλοιπα κομμάτια 30 δευτερολέπτων που έχουμε. Για να αποφύγουμε εισαγωγές που μπορούν να δίνουν την λάθος εντύπωση για το κομμάτι θα ελέγξουμε εάν το κομμάτι είναι πάνω από 1,5 λεπτό και εάν ναι θα λάβουμε το ηχητικό σήμα στο παράθυρο 0:40 - 1:10 αλλιώς στο παράθυρο 0:00 - 0:30.

Πρακτικά έχουμε $SAMPLE_RATE = 22050$, το ηχητικό σήμα ενός κομματιού στο 1:30 λεπτό είναι $22050 * 90 = 1984512$, έτσι θα ελέγξουμε πρώτα εάν το μήκος του κομματιού είναι πάνω από αυτόν τον αριθμό. Εάν ναι τότε θα πάρουμε το ηχητικό σήμα στο εύρος (0:40 - 1:10) δηλαδή $(40 * SAMPLE_RATE, 70 * SAMPLE_RATE) = (40 * 22050, 70 * 22050) = (882000, 1543504)$.

Αλλιώς θα πάρουμε το ηχητικό σήμα στο εύρος (0, 661504).

Όπως αναφέραμε παραπάνω, τα MFCC που θα εξαχθούν από ένα τραγούδι θα συγκριθούν με τα MFCC των υπόλοιπων τραγουδιών της βάσης με χρήση **Cosine Similarity** και **Euclidean Distance**. Το ένα τραγούδι περιέχει 1293 εγγραφές όπου η κάθε μια περιέχει 40 MFCCs. Ο αριθμός 1293 προκύπτει από $\frac{\text{DURATION} \cdot \text{SAMPLE_RATE}}{\text{hop_length}} = \frac{30 \cdot 22050}{512} = 1293$. Έτσι ένα τραγούδι αναπαρίσταται από έναν πίνακα (1283, 40). Αυτό που θα κάνουμε είναι να το μετατρέψουμε σε διάνυσμα με την μέθοδο `flatten`, έτσι θα μετατραπεί σε (51320, 1). Σε αυτή την μορφή θα μετατρέπουμε και όλα τα τραγούδια της βάσης για να μπορούμε να τα συγκρίνουμε.

Ο κώδικας είναι ο 11. Από όλα τα αποτελέσματα θα επιλέξουμε τα πρώτα 5 για να δείξουμε στον χρήστη.

6.5.2 Με χρήση άλλων Χαρακτηριστικών

Θα δημιουργήσουμε ένα ακόμη μοντέλο το οποίο δεν θα χρησιμοποιεί MFCC αλλά όλα τα χαρακτηριστικά τα οποία υπάρχουν μέσα στο csv αρχείο που έρχεται με το GTZAN framework, συγκεκριμένα σε αυτό το CSV αρχείο εμπεριέχονται χαρακτηριστικά για το κάθε κομμάτι που φαίνονται στον Πίνακα 2. Όλα τα παρακάτω αποθηκεύονται σε δύο μορφές, mean και variance και δεν είναι πίνακες αλλά μια τιμή.

Attribute	Description
Chroma STFT	Το χαρακτηριστικό chroma είναι ένας περιγραφέας, ο οποίος αναπαριστά το τονικό περιεχόμενο ενός μουσικού ηχητικού σήματος σε συμπυκνωμένη μορφή.
RMS	Έχει να κάνει με την ένταση του κομματιού.
Spectral Centroid	Το κέντρο του φάσματος του σήματος. Υπολογίζεται ως ο σταθμισμένος μέσος όρος των συχνοτήτων που υπάρχουν στο σήμα. Χαρακτηρίζει το ηχόχρωμα.
Spectral Bandwidth	Εύρος ζώνης είναι η διαφορά μεταξύ της ανώτερης και της κατώτερης συχνότητας σε μια συνεχή ζώνη συχνοτήτων.
Rolloff	Μπορεί να οριστεί ως η δράση ενός συγκεκριμένου τύπου φίλτρου το οποίο έχει σχεδιαστεί για να περιορίζει τις συχνότητες εκτός ενός συγκεκριμένου εύρους. Ο λόγος που το ονομάζουμε roll-off είναι επειδή πρόκειται για μια σταδιακή διαδικασία.
Zero crossing Rate	Ο αριθμός των τομών του άξονα χρόνου στο ηχητικό σήμα. Χαρακτηρίζει σήματα που μοιάζουν με θόρυβο.
Tempo	Ο ρυθμός του τραγουδιού.
MFCC [1-20]	Τα MFCC με 20 coefficients.

Πίνακας 2: Χαρακτηριστικά που θα εξαχθούν

Όταν ο χρήστης θα εισάγει ένα κομμάτι τα χαρακτηριστικά του Πίνακα 2 θα εξαχθούν από το κομμάτι και θα αποθηκευτούν σε ένα διάνυσμα, κάθε γραμμή του πίνακα θα είναι και

αυτή σε μορφή διανύσματος και θα συγκρίνονται μεταξύ τους πάλι με χρήση cosine similarity και EMD.

6.5.3 Με προσθήκη Metadata

Κάτι τελευταίο που θα προσθέσουμε στην προσέγγιση είναι το μοντέλο να λαμβάνει το αποτέλεσμα του Genre Classification και να αναζητά κομμάτια που αφορούν μόνο το συγκεκριμένο είδος. Αν για παράδειγμα το τραγούδι που εισάγεται αποφανθεί πως είναι Rock τότε ο αλγόριθμος θα εκτελεί cosine similarity με τραγούδια που ανήκουν στο γένος Rock και θα εμφανίζει τα πιο συναφή Rock κομμάτια σε σχέση με το κομμάτι εισόδου.

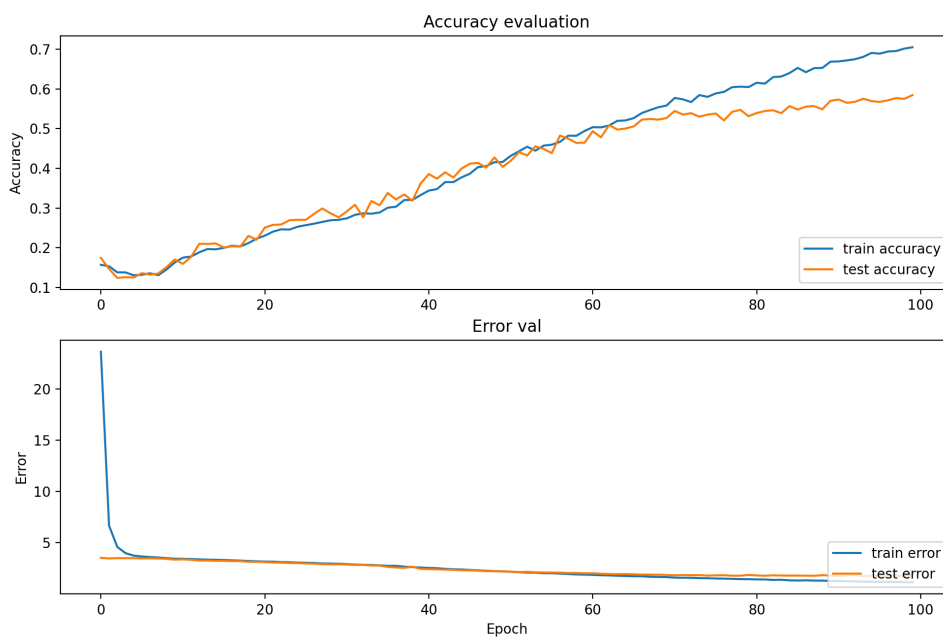
7 Πειραματικά Αποτελέσματα

7.1 Ταξινόμηση κομματιού σε Είδος

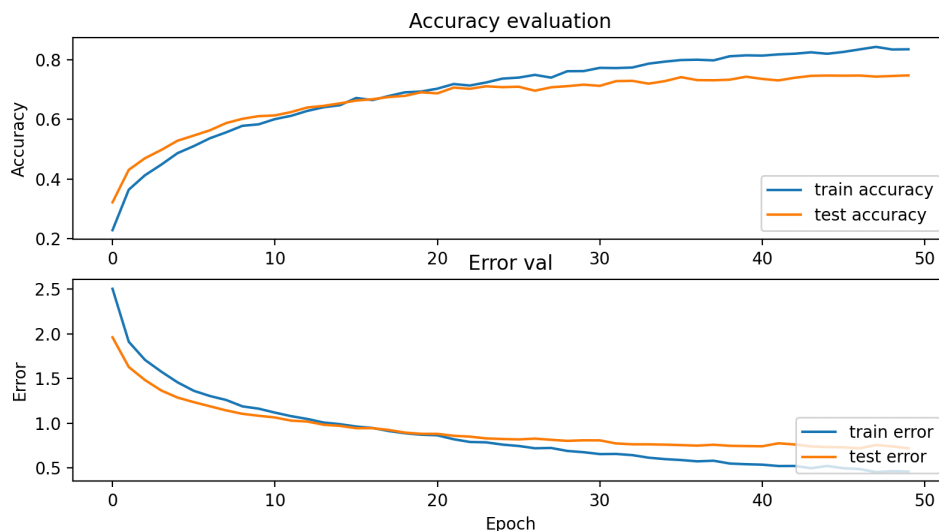
Όπως αναφέραμε και στο κεφάλαιο 6.4 οι παράμετροι που εφαρμόσαμε και στα 3 μοντέλα Νευρωνικών Δικτύων είναι παρόμοιοι. Τα δεδομένα εισόδου βρίσκονται σε μορφή MFCC με 13 coefficients, χρησιμοποιείται Sample Rate 22050, Hop Length 512, και το παράθυρο του FFT είναι 2048 samples. Η εκπαίδευση για το κάθε μοντέλο διαρκεί 50 epochs, εκπαιδεύεται πάνω στο 55% του Data, το 25% χρησιμοποιείται για Validation για το υπόλοιπο 20% για Test Data. Εφαρμόζουμε Mini-Batch προσέγγιση μεγέθους 32. Η αρχιτεκτονική του MLP μοντέλου φαίνεται στην **Εικόνα 54**, του CNN στην **Εικόνα 57** και τέλος του LSTM μοντέλου στην **Εικόνα 61**.

Model	Accuracy	Precision	Recall	F1-Score
MLP	0.614	0.629	0.614	0.614
CNN	0.723	0.717	0.723	0.718
LSTM	0.642	0.642	0.642	0.641

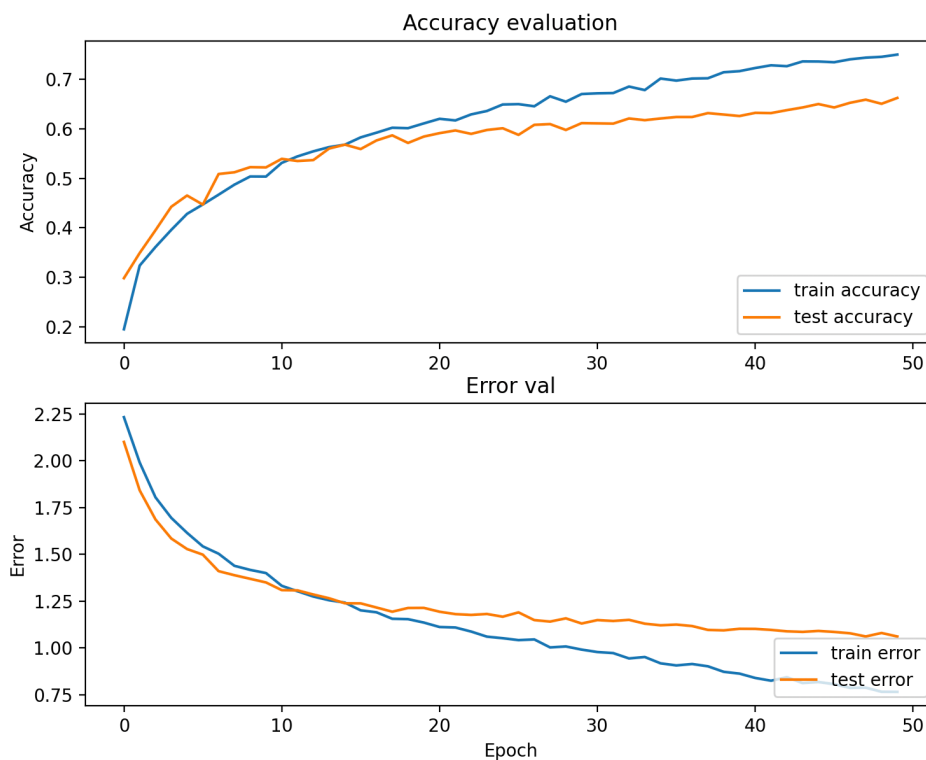
Πίνακας 3: Αποτελέσματα των Μοντέλων



Εικόνα 62: Multilayer Perceptron Network



Εικόνα 63: Convolutional Neural Network



Εικόνα 64: Long Short Term Memory Network

Από τα παραπάνω αποτελέσματα βλέπουμε ότι το μοντέλο με την καλύτερη απόδοση ήταν το CNN το οποίο ακολουθείται από το LSTM και τέλος είναι το MLP. Το LSTM δεν έχει μεγάλη διαφορά από το MLP αλλά το αρνητικό του ήταν ο πολύ μεγάλος σε έκταση χρόνος που χρειάστηκε για να εκπαιδευτεί. Οι διαφορές στις επιδόσεις μεταξύ των αρχιτεκτονικών νευρωνικών δικτύων για την ταξινόμηση μουσικών ειδών με χρήση MFCC μπορούν να αποδοθούν στα εγγενή πλεονεκτήματά τους:

- CNN

Χωρικά χαρακτηριστικά: Τα CNN υπερέχουν στην εξαγωγή χωρικών χαρακτηριστικών, γεγονός που τα καθιστά κατάλληλα για την ανάλυση μοτίβων συχνότητας σε MFCC. Η ικανότητά τους να μαθαίνουν αυτόματα ιεραρχικά χαρακτηριστικά βοηθά στην καταγραφή τόσο τοπικών όσο και πιο ευρέων μοτίβων στα μουσικά είδη.

- **LSTM**

Χρονική εστίαση: Οι LSTM έχουν σχεδιαστεί για τη σύλληψη χρονικών εξαρτήσεων σε διαδοχικά δεδομένα. Ωστόσο, για τα MFCC, όπου η χρονική πτυχή μπορεί να είναι λιγότερο κρίσιμη, οι LSTM ενδέχεται να μην αξιοποιούν πλήρως τις δυνατότητές τους.

- **MLP**

Επίπεδη δομή: Οι MLP δεν έχουν την ικανότητα να συλλάβουν χωρικές ή χρονικές εξαρτήσεις σε δεδομένα MFCC, εμποδίζοντας την ικανότητά τους να αναπαριστούν σύνθετες ιεραρχικές σχέσεις.

Η ταξινόμηση μουσικών ειδών περιλαμβάνει την αναγνώριση σύνθετων μοτίβων τόσο στο χρονικό όσο και στο φασματικό πεδίο. Η εξαγωγή χωρικών χαρακτηριστικών και η ιεραρχική μάθηση των CNN ευθυγραμμίζονται καλά με αυτές τις απαιτήσεις, οδηγώντας σε ανώτερες επιδόσεις. Συνολικά, η ικανότητα του CNN να συλλαμβάνει χωρικά χαρακτηριστικά και να μαθαίνει ιεραρχικές αναπαραστάσεις το καθιστά κατάλληλο για την ταξινόμηση μουσικών ειδών με χρήση MFCC, με αποτέλεσμα καλύτερες επιδόσεις σε σύγκριση με τις αρχιτεκτονικές LSTM και MLP.

Με τα παραπάνω ως συμπέρασμα στην τελική εφαρμογή που θα υλοποιήσουμε θα χρησιμοποιήσουμε το μοντέλο CNN για τον σκοπό του Music Genre Classification.

7.2 Σύσταση παρόμοιων κομματιών

Με την ανάλυση που κάναμε στο **Κεφάλαιο 6.5** θα επιχειρήσουμε δυο προσεγγίσεις, μια όπου όλα τα κομμάτια θα συγκρίνονται μεταξύ τους αποκλειστικά βάση των MFCC με αριθμό coefficients 40 και μια άλλη προσέγγιση όπου θα χρησιμοποιείται ο μέσος όρος από 20 MFCC και διάφορων άλλων μετρικών όπως Tempo, Spectral Flux, Chroma STFT, RMS, Zero Crossing Rate και άλλων. Επίσης η σύγκριση θα γίνεται με δύο τρόπους, με Cosine Similarity και Euclidean Metric Distance.

Με χρήση MFCC

Θα χρησιμοποιήσουμε μερικά γνωστά τραγούδια που κατεβάσαμε αλλά και κομμάτια μέσα από το Data Set για να δούμε τι θα μας προτείνει το μοντέλο. Ξεκινάμε με κλασική μουσική το Lacrimosa του Mozart και βλέπουμε ότι το μοντέλο μας προτείνει ορθά κλασική μουσική, με μια διαφοροποίηση στο EMD μοντέλο που μας προτείνει και ένα jazz κομμάτι.

```
Input path:
/Users/conspd/Downloads/Mozart-Lacrimosa.wav
Results with Cosine Similarity
Top 1 = classical.00087.wav, path /Users/conspd/School/Thesis/Data/genres_original/classical
Top 2 = classical.00084.wav, path /Users/conspd/School/Thesis/Data/genres_original/classical
Top 3 = classical.00011.wav, path /Users/conspd/School/Thesis/Data/genres_original/classical
Top 4 = classical.00083.wav, path /Users/conspd/School/Thesis/Data/genres_original/classical
Top 5 = classical.00028.wav, path /Users/conspd/School/Thesis/Data/genres_original/classical
Results with Euclidean Distance
```

```
Top with EMD1 = classical.00028.wav, path /Users/conspd/School/Thesis/Data/genres_original/classical
Top with EMD2 = classical.00068.wav, path /Users/conspd/School/Thesis/Data/genres_original/classical
Top with EMD3 = classical.00002.wav, path /Users/conspd/School/Thesis/Data/genres_original/classical
Top with EMD4 = classical.00025.wav, path /Users/conspd/School/Thesis/Data/genres_original/classical
Top with EMD5 = jazz.00057.wav, path /Users/conspd/School/Thesis/Data/genres_original/jazz
```

Έπειτα θα συνεχίσουμε με metal, Seek & Destroy των Metallica. Εδώ το CS (*Cosine Similarity*) μας προτείνει ορθά metal τραγούδια αλλά το EMD αποτυγχάνει και μας προτείνει country μουσική.

```
/Users/conspd/Downloads/Seek & Destroy (Remastered).mp3
Results with Cosine Similarity
Top 1 = metal.00072.wav, path /Users/conspd/School/Thesis/Data/genres_original/metal
Top 2 = metal.00083.wav, path /Users/conspd/School/Thesis/Data/genres_original/metal
Top 3 = metal.00022.wav, path /Users/conspd/School/Thesis/Data/genres_original/metal
Top 4 = metal.00075.wav, path /Users/conspd/School/Thesis/Data/genres_original/metal
Top 5 = metal.00023.wav, path /Users/conspd/School/Thesis/Data/genres_original/metal
Results with Euclidean Distance
Top with EMD1 = country.00001.wav, path /Users/conspd/School/Thesis/Data/genres_original/country
Top with EMD2 = country.00009.wav, path /Users/conspd/School/Thesis/Data/genres_original/country
Top with EMD3 = rock.00056.wav, path /Users/conspd/School/Thesis/Data/genres_original/rock
Top with EMD4 = rock.00053.wav, path /Users/conspd/School/Thesis/Data/genres_original/rock
Top with EMD5 = hiphop.00034.wav, path /Users/conspd/School/Thesis/Data/genres_original/hiphop
```

Μετά με Hip Hop από το Data Set το ίδιο.

```
/Users/conspd/School/Thesis/Data/genres_original/hiphop/hiphop.00083.wav
Results with Cosine Similarity
Top 1 = hiphop.00083.wav, path /Users/conspd/School/Thesis/Data/genres_original/hiphop
Top 2 = jazz.00018.wav, path /Users/conspd/School/Thesis/Data/genres_original/jazz
Top 3 = classical.00002.wav, path /Users/conspd/School/Thesis/Data/genres_original/classical
Top 4 = hiphop.00082.wav, path /Users/conspd/School/Thesis/Data/genres_original/hiphop
Top 5 = hiphop.00090.wav, path /Users/conspd/School/Thesis/Data/genres_original/hiphop
Results with Euclidean Distance
Top with EMD1 = hiphop.00083.wav, path /Users/conspd/School/Thesis/Data/genres_original/hiphop
Top with EMD2 = metal.00093.wav, path /Users/conspd/School/Thesis/Data/genres_original/metal
Top with EMD3 = hiphop.00090.wav, path /Users/conspd/School/Thesis/Data/genres_original/hiphop
Top with EMD4 = hiphop.00082.wav, path /Users/conspd/School/Thesis/Data/genres_original/hiphop
Top with EMD5 = metal.00007.wav, path /Users/conspd/School/Thesis/Data/genres_original/metal
```

Μετά με Pop από το Data Set το ίδιο. Βλέπουμε ότι μας προτείνει κομμάτια τα οποία δεν ανήκουν στο είδος που αναφέρεται, ωστόσο ακούγοντας τα βλέπουμε ότι έχουν παρόμοια χαρακτηριστικά με το αρχικό κομμάτι αλλά είναι διαφορετικό στυλ.

```
/Users/conspd/School/Thesis/Data/genres_original/pop/pop.00092.wav
Results with Cosine Similarity
Top 1 = pop.00092.wav, path /Users/conspd/School/Thesis/Data/genres_original/pop
Top 2 = jazz.00064.wav, path /Users/conspd/School/Thesis/Data/genres_original/jazz
Top 3 = metal.00097.wav, path /Users/conspd/School/Thesis/Data/genres_original/metal
Top 4 = metal.00030.wav, path /Users/conspd/School/Thesis/Data/genres_original/metal
Top 5 = pop.00040.wav, path /Users/conspd/School/Thesis/Data/genres_original/pop
Results with Euclidean Distance
Top with EMD1 = pop.00092.wav, path /Users/conspd/School/Thesis/Data/genres_original/pop
Top with EMD2 = rock.00056.wav, path /Users/conspd/School/Thesis/Data/genres_original/rock
Top with EMD3 = country.00001.wav, path /Users/conspd/School/Thesis/Data/genres_original/country
Top with EMD4 = country.00000.wav, path /Users/conspd/School/Thesis/Data/genres_original/country
Top with EMD5 = pop.00004.wav, path /Users/conspd/School/Thesis/Data/genres_original/pop
```

Με χρήση άλλων Χαρακτηριστικών

```
Input Path:
/Users/conspd/Downloads/Mozart-Lacrimosa.wav
Top 1 = classical.00050.wav
Top 2 = classical.00085.wav
Top 3 = classical.00073.wav
Top 4 = classical.00042.wav
Top 5 = classical.00055.wav
Results with Euclidean Distance
Top EMD1 = jazz.00070.wav
Top EMD2 = classical.00060.wav
Top EMD3 = country.00065.wav
Top EMD4 = classical.00019.wav
Top EMD5 = rock.00093.wav
```

```

/Users/conspd/Downloads/Seek & Destroy (Remastered).mp3
Top 1 = rock.00097.wav
Top 2 = disco.00086.wav
Top 3 = country.00092.wav
Top 4 = country.00033.wav
Top 5 = disco.00009.wav
Results with Euclidean Distance
Top EMD1 = disco.00058.wav
Top EMD2 = metal.00088.wav
Top EMD3 = jazz.00084.wav
Top EMD4 = country.00096.wav
Top EMD5 = country.00095.wav

```

```

/Users/conspd/School/Thesis/Data/genres_original/hiphop/hiphop.00083.wav
Top 1 = hiphop.00083.wav
Top 2 = disco.00090.wav
Top 3 = rock.00080.wav
Top 4 = disco.00014.wav
Top 5 = hiphop.00068.wav
Results with Euclidean Distance
Top EMD1 = hiphop.00083.wav
Top EMD2 = blues.00070.wav
Top EMD3 = rock.00053.wav
Top EMD4 = country.00062.wav
Top EMD5 = metal.00083.wav

```

```

/Users/conspd/School/Thesis/Data/genres_original/pop/pop.00092.wav
Top 1 = pop.00092.wav
Top 2 = rock.00012.wav
Top 3 = disco.00048.wav
Top 4 = country.00095.wav
Top 5 = metal.00086.wav
Results with Euclidean Distance
Top EMD1 = pop.00092.wav
Top EMD2 = reggae.00080.wav
Top EMD3 = pop.00008.wav
Top EMD4 = rock.00012.wav
Top EMD5 = pop.00028.wav

```

Μπορούμε να εξάγουμε σαν συμπέρασμα ότι η προσέγγιση με τα 40 MFCC διατηρώντας όλα τα στοιχεία και χωρίς να εκτελεί τον μέσο όρο τους ήταν καλύτερη και αυτή είναι που θα διατηρήσουμε στο τελικό πρόγραμμα. Καθώς επίσης και ότι το Cosine Similarity επέστρεφε πιο συναφή τραγούδια σε σχέση με το EMD.

Τέλος προσθέτουμε και την metadata προσέγγιση στο Recommendation Engine όπου το Music Genre Classification λειτουργεί συνεργατικά με το Recommendation Engine με τον τρόπο ότι το Recommendation Engine δέχεται το αποτέλεσμα που εξάγει το CNN όσον αφορά το είδος μουσικής του τραγουδιού που εισάγεται και ψάχνει παρόμοια τραγούδια του συγκεκριμένου είδους.

```

Give path of song:
/Users/conspd/Downloads/Songs/Mozart-Lacrimosa.wav
Top 1 = classical.00087.wav, path /Users/conspd/School/Thesis/Data/genres_original/classical
Top 2 = classical.00084.wav, path /Users/conspd/School/Thesis/Data/genres_original/classical
Top 3 = classical.00011.wav, path /Users/conspd/School/Thesis/Data/genres_original/classical
Top 4 = classical.00083.wav, path /Users/conspd/School/Thesis/Data/genres_original/classical
Top 5 = classical.00028.wav, path /Users/conspd/School/Thesis/Data/genres_original/classical

```

```

Give path of song:
/Users/conspd/Downloads/Songs/Seek & Destroy (Remastered).mp3
Top 1 = metal.00072.wav, path /Users/conspd/School/Thesis/Data/genres_original/metal
Top 2 = metal.00083.wav, path /Users/conspd/School/Thesis/Data/genres_original/metal
Top 3 = metal.00022.wav, path /Users/conspd/School/Thesis/Data/genres_original/metal
Top 4 = metal.00075.wav, path /Users/conspd/School/Thesis/Data/genres_original/metal
Top 5 = metal.00023.wav, path /Users/conspd/School/Thesis/Data/genres_original/metal

```

```

Give path of song:
/Users/conspd/School/Thesis/Data/genres_original/hiphop/hiphop.00083.wav
Top 1 = hiphop.00083.wav, path /Users/conspd/School/Thesis/Data/genres_original/hiphop
Top 2 = hiphop.00082.wav, path /Users/conspd/School/Thesis/Data/genres_original/hiphop
Top 3 = hiphop.00090.wav, path /Users/conspd/School/Thesis/Data/genres_original/hiphop
Top 4 = hiphop.00096.wav, path /Users/conspd/School/Thesis/Data/genres_original/hiphop
Top 5 = hiphop.00085.wav, path /Users/conspd/School/Thesis/Data/genres_original/hiphop

```

```
Give path of song:  
/Users/conspd/School/Thesis/Data/genres_original/pop/pop.00092.wav  
Top 1 = pop.00092.wav, path /Users/conspd/School/Thesis/Data/genres_original/pop  
Top 2 = pop.00040.wav, path /Users/conspd/School/Thesis/Data/genres_original/pop  
Top 3 = pop.00004.wav, path /Users/conspd/School/Thesis/Data/genres_original/pop  
Top 4 = pop.00069.wav, path /Users/conspd/School/Thesis/Data/genres_original/pop  
Top 5 = pop.00017.wav, path /Users/conspd/School/Thesis/Data/genres_original/pop
```

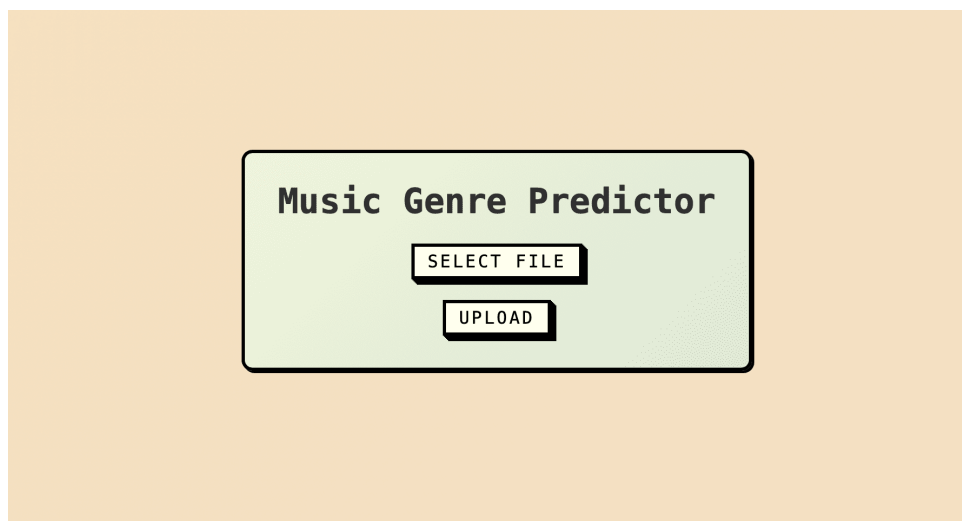
Εν κατακλείδι η βέλτιστη προσέγγιση για το Recommendation Engine είναι αυτή που:

1. Εξάγει από το κομμάτι 40 MFCC Coefficients και τα συγκρίνει με τα MFCC των άλλων κομματιών
2. Η σύγκριση εκτελείται με χρήση του Cosine Similarity
3. Χρησιμοποιεί και ένα Metadata το οποίο είναι το είδος μουσικής του κομματιού που εξάγεται σαν αποτέλεσμα του Music Genre Classification και με αυτόν τον τρόπο συγκρίνει κομμάτια του ίδιου είδους

Τα παραπάνω που αναφερθήκαν θα είναι αυτά που θα αποτελούν τα συστατικά του Recommendation Engine του τελικού μοντέλου της εφαρμογής.

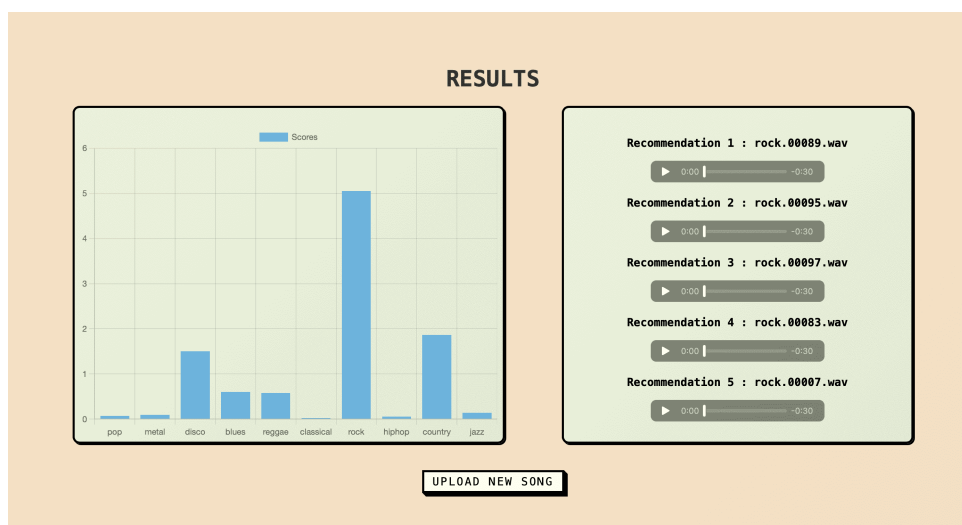
8 Εκτέλεση της Τελικής Εφαρμογής

Σε αυτό το κομμάτι θα δείξουμε την λειτουργία της εφαρμογής μαζί με το Web Interface που δημιουργήσαμε χρησιμοποιώντας το framework Flask.

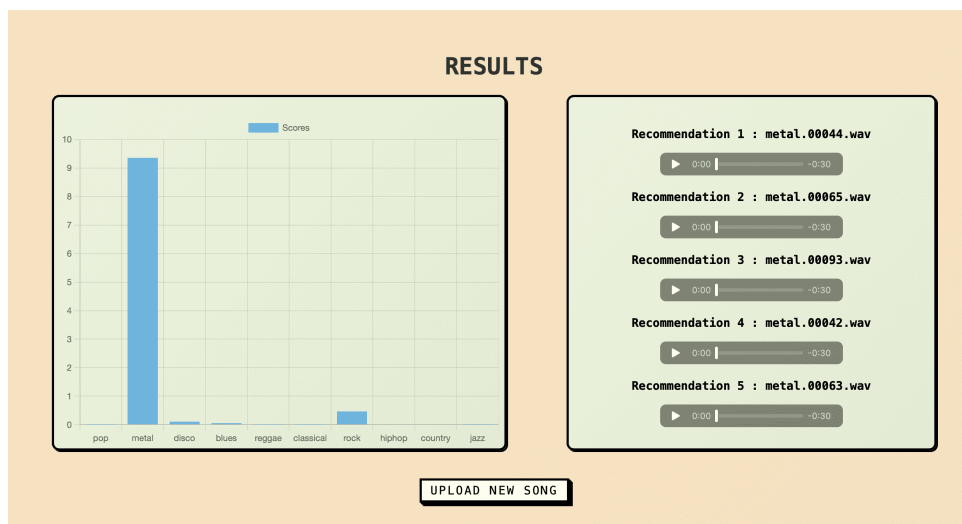


Εικόνα 65: Home Page της Εφαρμογής

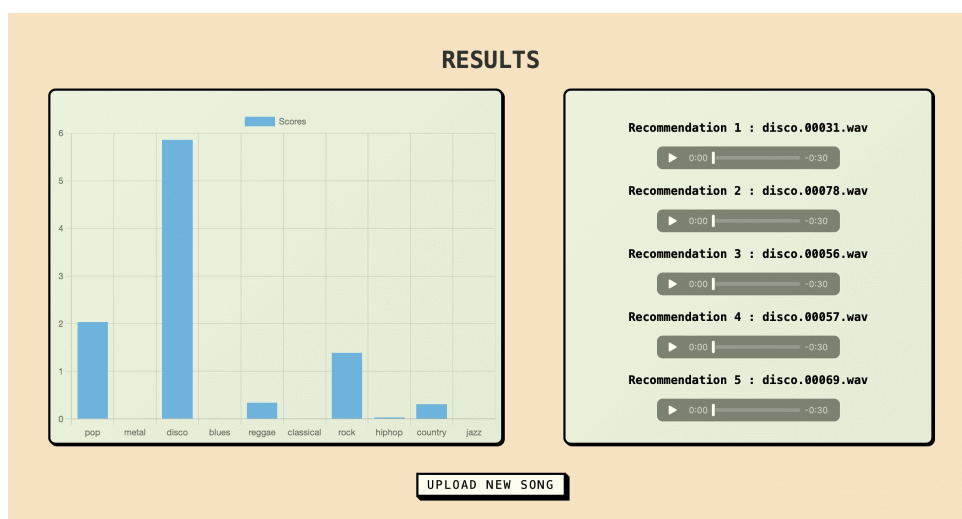
8.1 Εισαγωγή Κομματιών από το Dataset



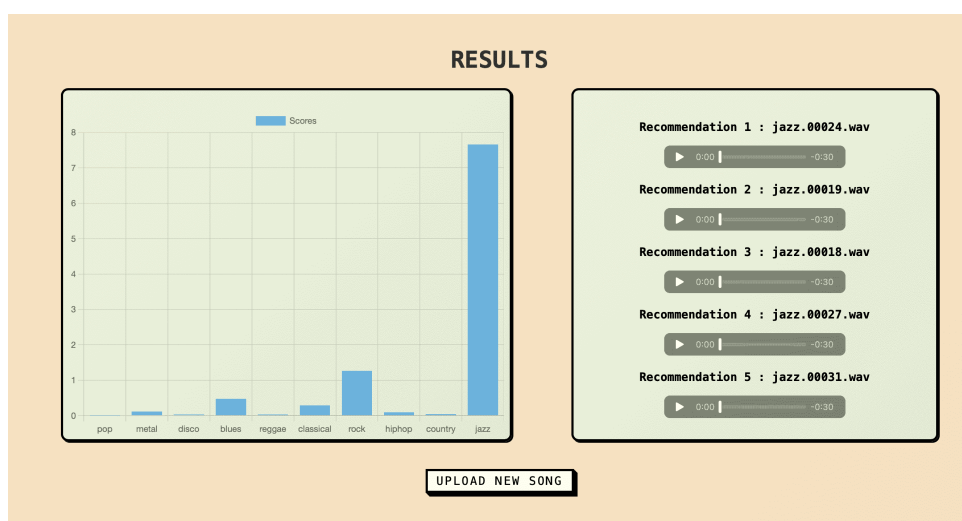
Εικόνα 66: Κομμάτι: rock.00089.wav



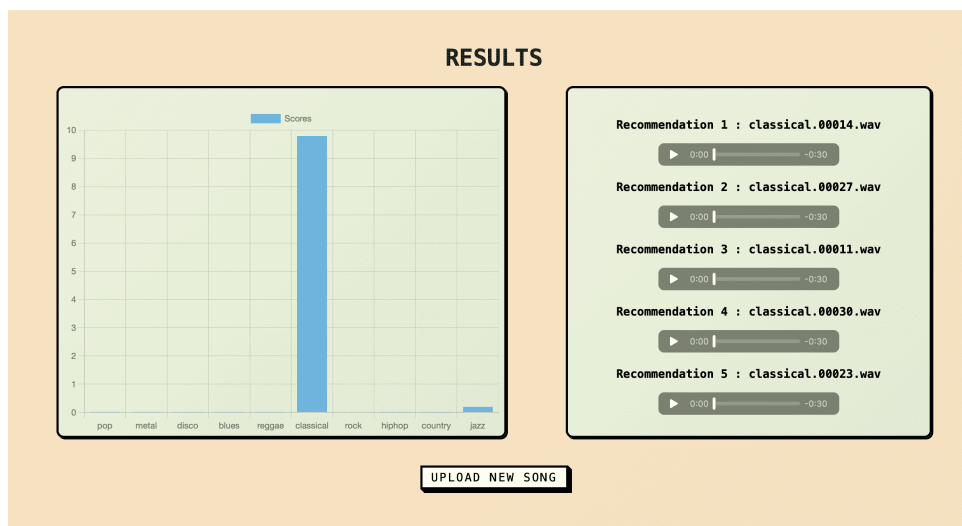
Εικόνα 67: Κομμάτι: metal.00065.wav



Εικόνα 68: Κομμάτι: disco.00031.wav



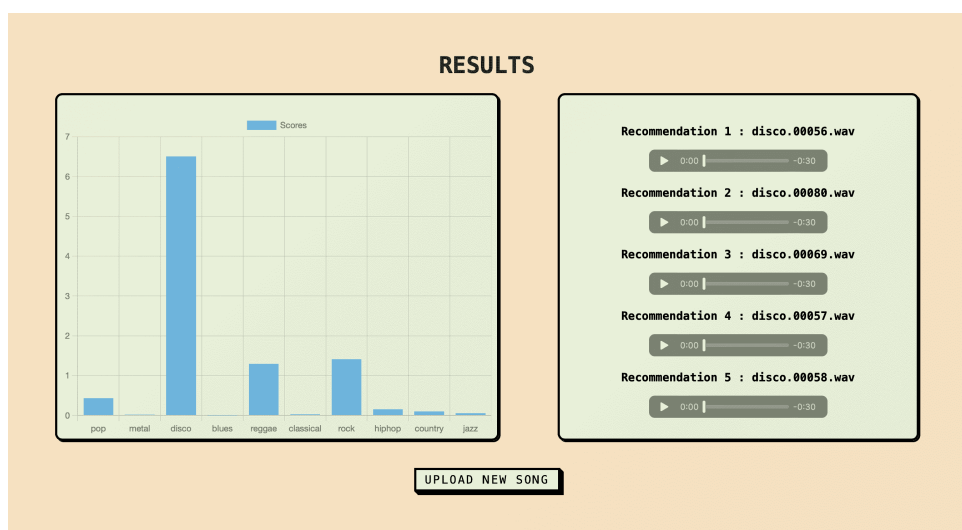
Εικόνα 69: Κομμάτι: jazz.00024.wav



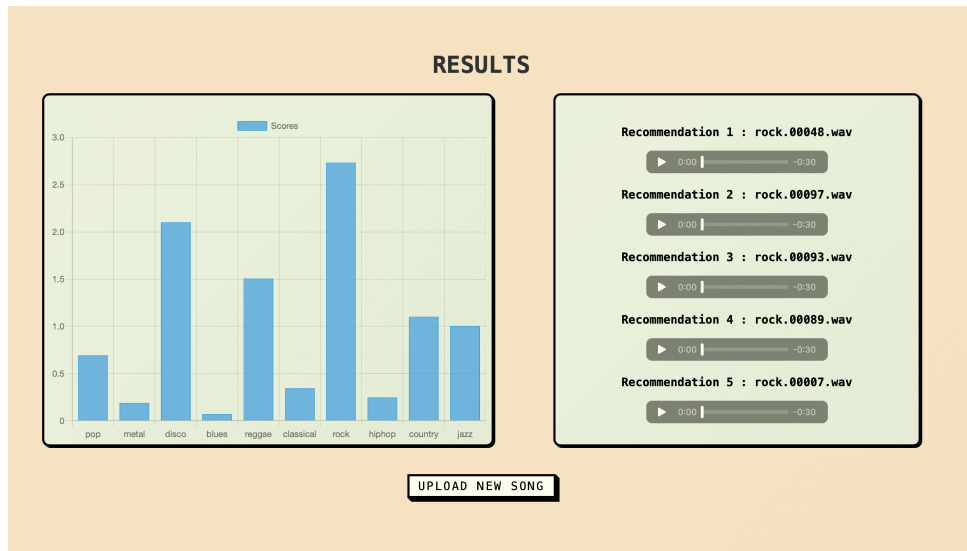
Εικόνα 70: Κομμάτι: classical.00014.wav

Από ότι βλέπουμε από τα λίγα παραδείγματα που δείξαμε το μοντέλο έχει άριστη απόδοση σε κομμάτια που ήδη έχει δει.

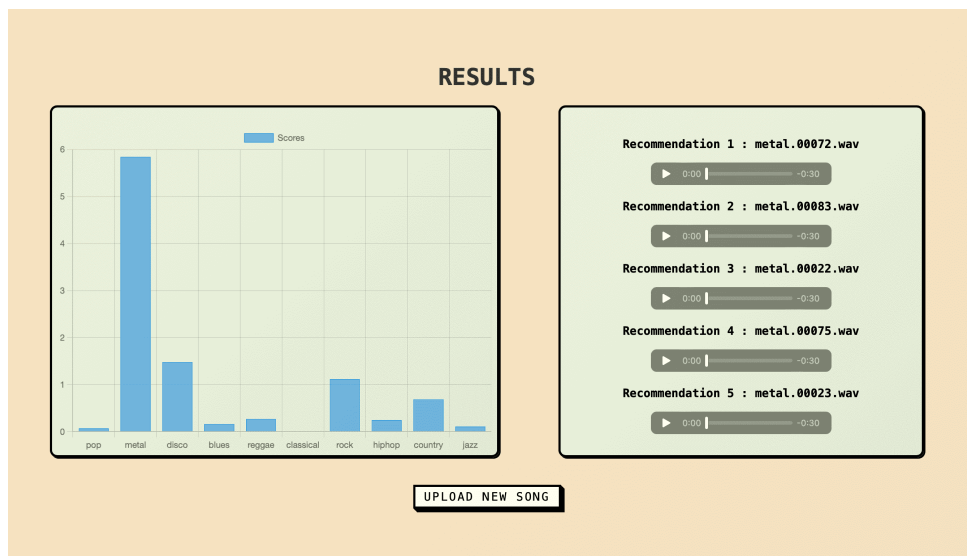
8.2 Εισαγωγή Κομματιών εκτός του Dataset



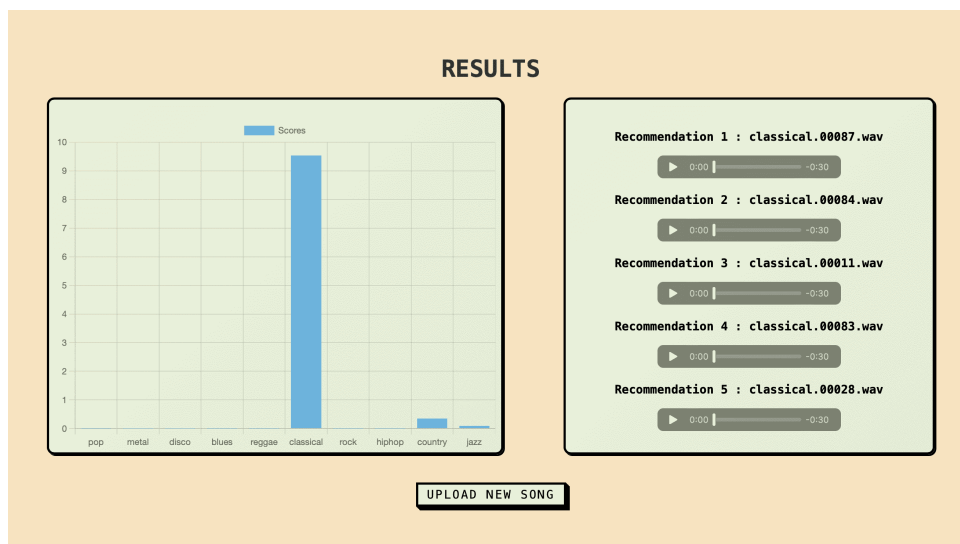
Εικόνα 71: Κομμάτι: Earth Wind & Fire – Boogie Wonderland



Εικόνα 72: Κομμάτι: The Doors – Roadhouse Blues



Εικόνα 73: Κομμάτι: Metallica – Seek and Destroy



Εικόνα 74: Κομμάτι: Mozart – Lacrimosa

Αυτά τα κομμάτια εισήχθησαν ολόκληρα σε μορφή .wav ή σε .mp3 και παρήχθησαν ικανοποιητικά αποτελέσματα με εξαίρεση το κομμάτι του Ice Cube το οποίο πιάνεται κανονικά hip hop και όχι reggae.

9 Συμπεράσματα & Μελλοντικές Επεκτάσεις

Στο κεφάλαιο αυτό ολοκληρώνει η διπλωματική εργασία, θα αναφερθούμε συνοπτικά στους τομείς που δουλέψαμε και αναλύσαμε, τις τεχνολογίες που χρησιμοποιήσαμε και τα συμπεράσματα που εξάγαμε.

Σε αυτή την εργασία ασχοληθήκαμε με τον τομέα του Music Information Retrieval και πιο συγκεκριμένα την ταξινόμηση μουσικών ειδών και συστήματα συστάσεων που αποτελούν υποτομείς του.

Ερευνήσαμε τις μετρικές οι οποίες χαρακτηρίζουν καλύτερα ένα μουσικό κομμάτι σε μορφή που μπορεί να διαβάσει και να εκπαιδευτεί ένας υπολογιστής όπου η πιο αποδοτική μετρική και για τον τομέα του MGC αλλά και των Recommendation Systems ήταν τα Mel-Frequency Cepstral Coefficients.

Μέσω βιβλιογραφικής ανασκόπησης **Κεφάλαιο 2** και πρακτικής εφαρμογής των Multilayer Perceptron, Convolutional Neural Network και Long-Short Term Memory νευρωνικών δικτύων **Κεφάλαιο 6.4** καταλήξαμε στο CNN να είναι το πιο αποδοτικό με ακρίβεια **0.723%** ακολουθούμενο από το LSTM με **0.642%** και τέλος το MLP με ακρίβεια **0.614%**.

Στο κομμάτι του Recommendation Engine εφαρμόσαμε content-based recommendations όπου το κάθε κομμάτι που εισάγεται από τον χρήστη συγκρίνεται με τα υπόλοιπα με βάση τα χαρακτηριστικά τους και όχι με βάση Metadata. Για την σύγκριση ακολουθήσαμε 2 προσεγγίσεις, μια συγκρίνοντας με 40 MFCC coefficients μεταξύ του κάθε κομματιού και μια άλλη με άλλου τύπου μετρικές όπως Chroma STFT, RMS, Spectral Centroid, Rolloff, Tempo, Zero crossing Rate, η σύγκριση με βάση των 40 MFCC coefficients αποδείχθηκε να είναι αυτή που εξάγει τις πιο συναφείς συστάσεις. Για τον αλγόριθμο σύγκρισης εφαρμόσαμε Cosine Similarity και Euclidean Distance όπου το Cosine Similarity επικράτησε παρουσιάζοντας πιο καλά αποτελέσματα. Τέλος προσθέσαμε τον συνδυαστικό κρίκο μεταξύ του Genre Classification και του Recommendation Engine το οποίο ήταν πρώτα να εξάγει ο αλγόριθμος τι είδος είναι το κομμάτι που εισάγεται και έπειτα να γίνεται αναζήτηση παρόμοιου κομματιού στο συγκεκριμένο είδος μουσικής.

Όλα τα παραπάνω που αναφέραμε τα υλοποιήσαμε σε Python με χρήση της βιβλιοθήκης Tensorflow για την δημιουργία του Νευρωνικού Δικτύου, Librosa για την εξαγωγή πληροφορίας από τα κομμάτια και Flask για να δημιουργήσουμε ένα απλό Web App το οποίο θα αναδεικνύει την λειτουργία της εφαρμογής. Για την εκπαίδευση του Δικτύου και του Συστήματος Συστάσεων χρησιμοποιήσαμε το GTZAN dataset το οποίο αποτελεί ένα από τα πιο γνωστά και βασικά Datasets στο κομμάτι του Music Genre Classification.

Μελλοντικές Επεκτάσεις

Μια χρήση της εφαρμογής που υλοποιήσαμε πέρα από την αναγνώριση ενός κομματιού και σύστασης ενός παρόμοιου κομματιού με αυτό μπορεί να είναι να εισαχθεί μια ολόκληρη βάση από κομμάτια προς ταξινόμηση και να δημιουργεί φακέλους ανάλογα το στυλ του κομματιού ώστε σε κάθε φάκελο - genre να εισάγονται κομμάτια που αντιστοιχούν σε αυτό. Έτσι π.χ. τα 1000 κομμάτια χωρίς label ταξινομούνται σε φακέλους ανάλογα με το είδος τους. Έπειτα αφού ταξινομηθούν θα εισάγουμε ένα κομμάτι και θα μας προτείνει παρόμοια κομμάτια με

αυτό σύμφωνα με τα καινούρια κομμάτια που εισάγαμε.

Η εφαρμογή μπορεί να επεκταθεί εύκολα εισάγοντας κι άλλα είδη μουσικής απλά δημιουργώντας καινούριους φακέλους και ονομάζοντας τους με το *genre* που θέλουμε να δημιουργήσουμε και μέσα σε αυτούς να εισάγουμε 100 ανάλογα κομμάτια. Έτσι κατεβάζοντας λίστες από συγκεκριμένα είδη μουσικής μπορούμε στην θεωρία να δημιουργήσουμε ένα μοντέλο που αναγνωρίζει 100 διαφορετικά είδη μουσικής και να το κάνουμε ακόμη πιο αποδοτικό με το να αυξήσουμε τον αριθμό των κομματιών του κάθε είδους.

Το παραπάνω μπορεί να επεκταθεί στο να δημιουργούμε δικές μας λίστες όχι απαραίτητα με το είδος μουσικής αλλά το συναίσθημα της μουσικής, π.χ. να έχουμε 10 φακέλους που αφορούν συναίσθημα όπως *happy*, *sad*, *motivational* ... και να εκπαιδεύσουμε το μοντέλο να αναγνωρίζει το συναίσθημα που βγάζει κάποιο κομμάτι και να το εκπαιδεύσουμε να μας προτείνει παρόμοια κομμάτια. Όλη η λογική αυτή και η επεκτάσεις της έγκειται στο να χρησιμοποιήσουμε το παρόν μοντέλο και να του εισάγουμε μια καινούρια βάση δεδομένων.

10 Παράρτημα

10.1 Τελικός παραδοτέος Κώδικας

Κώδικας 2: Τελικός κώδικας του Web Interface

```

1 from flask import Flask, render_template, request
2 import os
3 import numpy as np
4 import keras
5 import librosa, librosa.feature
6 from werkzeug.utils import secure_filename
7 import json
8 from scipy.spatial.distance import cosine
9
10 DATASET_PATH = "MGC/data.json"
11 DATASET_PATH_RS = "Recommender Systems/recommendation_data.json"
12 SAMPLES_PER_TRACK = 22050 * 30
13 num_samples_per_segment = int(SAMPLES_PER_TRACK / 10)
14 UPLOAD_FOLDER = "/Users/consdp/Desktop/Python/uploads"
15 app = Flask(__name__)
16 app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
17
18 @app.route('/')
19 @app.route('/index')
20 def index():
21     return render_template('index.html')
22
23 @app.route('/songupload', methods=['POST'])
24 def song_upload():
25     file = request.files['audio']
26     filename = secure_filename(file.filename)
27     file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
28
29     song, mfcc_rs = load_song(os.path.join(app.config['UPLOAD_FOLDER'], filename))
30     return predict(model, song, mfcc_rs)
31
32 def find_most_similar_song(song, list_of_songs, predicted_genre):
33     cosine_similarity_score = []
34     i = 0
35     for s in list_of_songs:
36         if predicted_genre in song_metadata["name"][i]:
37             cosine_similarity_score.append(1 - cosine(song.flatten(), s.flatten()))
38         else:
39             cosine_similarity_score.append(-1)
40         i = i + 1
41     top_indices = sorted(enumerate(cosine_similarity_score), key=lambda x: x[1],
42 reverse=True)[:5]
43     top_indices_only = [index for index, _ in top_indices]
44     return top_indices_only
45
46 def load_song(song_path):
47     signal, sr = librosa.load(song_path, sr=22050)

```



```

47
48 # For MGC
49 mfcc_table = []
50 for s in range(10):
51     start_sample = num_samples_per_segment * s
52     finish_sample = start_sample + num_samples_per_segment
53     mfcc = librosa.feature.mfcc(y=signal[start_sample:finish_sample], n_mfcc=13,
54     n_fft=2048, hop_length=512).T
55     mfcc = mfcc[..., np.newaxis]
56     mfcc_table.append(mfcc)
57
58 # For RS
59 if len(signal) > 1984512:
60     mfcc_rs = librosa.feature.mfcc(y=signal[882000:1543504], sr=sr, n_mfcc=40,
61     n_fft=2048, hop_length=512).T
62 else:
63     mfcc_rs = librosa.feature.mfcc(y=signal[:661504], sr=sr, n_mfcc=40, n_fft
64     =2048, hop_length=512).T
65
66 return mfcc_table, mfcc_rs
67
68 def predict(model, song, mfcc_rs):
69     sum_preds = np.zeros(10)
70     for s in song:
71         s = s[np.newaxis, ...]
72         prediction = model.predict(s)
73         preds = np.asarray(prediction[0])
74         sum_preds = np.add(sum_preds, preds)
75     genres = ["pop", "metal", "disco", "blues", "reggae", "classical", "rock", "
76     hiphop", "country", "jazz"]
77     predicted_genre = np.argmax(sum_preds) # Genre of Music that was mostly
78     predicted
79     index = find_most_similar_song(mfcc_rs, list_of_songs, genres[predicted_genre])
80     list_of_paths = []
81     list_of_names = []
82     for k, i in enumerate(index):
83         # list_of_paths.append(song_metadata["path"][i]+'/'+song_metadata["name"][i
84         ])
85         list_of_paths.append(song_metadata["path"][i] + '/' + song_metadata["name"][
86         i])
87         list_of_names.append(song_metadata["name"][i])
88
89     return render_template('showresults.html', sum_preds=sum_preds, genres=genres,
90     paths=zip(list_of_paths, list_of_names))
91
92 if __name__ == "__main__":
93     # Load the Model
94     model = keras.models.load_model('MGC/mymodel.keras')
95
96     # Load data for RS
97     with open(DATASET_PATH_RS, "r") as fp:

```

```
91     data = json.load(fp)
92     list_of_songs = np.array(data["mfcc"])
93     song_metadata = {"name": np.array(data["name"]), "path": np.array(data["path"])}
94
95     app.run(debug=True)
```

Κώδικας 3: Home Page

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Music Genre Predictor</title>
7   <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='
  styles/style.css')}}">
8 </head>
9 <body>
10  <div class="d-flex flex-column justify-content-center w-100 h-100">
11    <div class="box">
12      <h1>Music Genre Predictor</h1>
13      <form action="/songupload" method="POST" enctype="multipart/form-data">
14        <input type="file" style="display:none;" id="song" name="audio"
  accept=".wav,.mp3">
15        <label for="song" class="button-54">Select File</label>
16        <br>
17        <input type="submit" class="button-54" value="Upload">
18      </form>
19      <div id="result"></div>
20    </div>
21  </div>
22 </body>
23 </html>

```

Κώδικας 4: Results Page

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Title</title>
6   <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
7   <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='
  styles/style.css')}}">
8 </head>
9
10 <body>
11  <h1>RESULTS</h1>
12  <div style="width: 80%; display: table;">
13    <div style="display: table-row; height: 100px;">
14      <div style="width: 50%; display: table-cell; vertical-align: bottom;"
  class="box">
15        <canvas id="barChart" width="600" height="450"></canvas>
16      </div>
17      <div class="cellContainer">
18        <div style="margin: 15px;"></div>
19      </div>
20      <div style="display: table-cell; vertical-align: middle; padding:25px"
  class="box">
21        {% for s in paths %}

```

```

22         <div><p><strong>Recommendation {{ loop.index0 + 1 }} : {{ s[1]
}} </strong></p></div>
23         <audio controls>
24             <source src="{{ url_for('static', filename=s[0][32:]) }}"
type="audio/wav">
25             Your browser does not support the audio element.
26         </audio>
27         {% endfor %}
28     </div>
29 </div>
30 </div>
31 <form action="/index">
32     <br><br>
33     <input type="submit" value="Upload New Song" class="button-54">
34 </form>
35 <script>
36     document.addEventListener('DOMContentLoaded', function () {
37         var ctx = document.getElementById('barChart').getContext('2d');
38
39         var data = {
40             labels: [{% for genre in genres %}"{{ genre }}"{% if not loop.last
%},{% endif %}{% endfor %}],
41             datasets: [{
42                 label: 'Scores',
43                 data: [{% for score in sum_preds %}{{ score }}{% if not loop.
last %},{% endif %}{% endfor %}],
44                 backgroundColor: 'rgba(52, 152, 219, 0.7)', // Adjust the color
as needed
45                 borderColor: 'rgba(52, 152, 219, 1)',
46                 borderWidth: 1
47             }]
48         };
49
50         var options = {
51             responsive: false, // Set responsive to false
52             scales: {
53                 y: {
54                     beginAtZero: true
55                 }
56             }
57         };
58
59         var myChart = new Chart(ctx, {
60             type: 'bar',
61             data: data,
62             options: options
63         });
64     });
65 </script>
66 </body>
67 </html>

```

Κώδικας 5: Προεπεξεργασία των Δεδομένων

```

1 import os
2 import librosa
3 import librosa.feature
4 import math
5 import json
6
7 DATASET_PATH = "/Users/conspd/School/Thesis/Data/genres_original"
8 # JSON_PATH = "/Users/conspd/Desktop/data1.json"
9 JSON_PATH = "data.json"
10 SAMPLE_RATE = 22050
11 DURATION = 30 # seconds
12 SAMPLES_PER_TRACK = SAMPLE_RATE * DURATION
13
14 def save_mfcc(dataset_path, json_path, n_mfcc=13, n_fft=2048, hop_length=512,
15             num_segments=5):
16     # build dictionary to store data
17     data = {
18         "mapping": [],
19         "mfcc": [],
20         "labels": []
21     }
22
23     num_samples_per_segment = int(SAMPLES_PER_TRACK / num_segments)
24     expected_num_mfcc_vectors_per_segment = math.ceil(num_samples_per_segment /
25             hop_length)
26
27     # loop through all the genres
28     for i, (dirpath, dirnames, filenames) in enumerate(os.walk(dataset_path)):
29         # ensure that we're not at the root level
30         if dirpath is not dataset_path:
31
32             # save the semantic label
33             dirpath_components = dirpath.split("/")
34             semantic_label = dirpath_components[-1]
35             data["mapping"].append(semantic_label)
36             print(f"\nProcessing {semantic_label}")
37
38             # process files for a specific genre
39             for f in filenames:
40                 file_path = os.path.join(dirpath, f)
41                 signal, sr = librosa.load(file_path, sr=SAMPLE_RATE)
42
43                 # process segments extracting mfcc and storing data
44                 for s in range(num_segments):
45                     start_sample = num_samples_per_segment * s
46                     finish_sample = start_sample + num_samples_per_segment
47
48                     mfcc = librosa.feature.mfcc(y=signal[start_sample:finish_sample
49 ],
50
51                                     sr=sr,
52                                     n_mfcc=n_mfcc,

```

```
49         n_fft=n_fft,
50         hop_length=hop_length)
51     mfcc = mfcc.T
52
53     # Store mfcc for segment if it has the expected length
54     if len(mfcc) == expected_num_mfcc_vectors_per_segment:
55         data["mfcc"].append(mfcc.tolist())
56         data["labels"].append(i-1)
57         # print(f"{file_path}, segment:{s+1}")
58
59     with open(JSON_PATH, "w") as fp:
60         json.dump(data, fp, indent=4)
61
62
63 if __name__=="__main__":
64     save_mfcc(DATASET_PATH, JSON_PATH, num_segments=10)
```

Κώδικας 6: Εξαγωγή MFCC για RS

```

1 import os
2 import librosa
3 import librosa.feature
4 import json
5
6
7 DATASET_PATH = "/Users/conspd/School/Thesis/Data/genres_original"
8 JSON_PATH = "recommendation_data.json"
9
10
11 def save_mfcc(dataset_path, json_path, n_mfcc=40, n_fft=2048, hop_length=512):
12     data = {
13         "name": [],
14         "mfcc": [],
15         "path": [],
16     }
17
18     for i, (dirpath, dirnames, filenames) in enumerate(os.walk(dataset_path)):
19         if dirpath is not dataset_path:
20             dirpath_components = dirpath.split("/")
21             semantic_label = dirpath_components[-1]
22
23             print(f"\nProcessing {semantic_label}")
24
25             for f in filenames:
26                 file_path = os.path.join(dirpath, f)
27                 signal, sr = librosa.load(file_path, sr=22050)
28
29                 # process segments extracting mfcc and storing data
30                 mfcc = librosa.feature.mfcc(y=signal[:661504],
31                                             sr=22050,
32                                             n_mfcc=n_mfcc,
33                                             n_fft=n_fft,
34                                             hop_length=hop_length)
35
36                 mfcc = mfcc.T
37                 if len(mfcc) == 1293:
38                     data["mfcc"].append(mfcc.tolist())
39                     data["name"].append(f)
40                     data["path"].append(dirpath)
41
42             with open(JSON_PATH, "w") as fp:
43                 json.dump(data, fp, indent=4)
44
45 if __name__ == "__main__":
46     save_mfcc(DATASET_PATH, JSON_PATH)

```

10.2 Λοιποί Κώδικες

Κώδικας 7: Δημιουργία και Εκπαίδευση Μοντέλου MLP

```

1 import json
2 import numpy as np
3 import keras
4 from sklearn.model_selection import train_test_split
5 import matplotlib.pyplot as plt
6 from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
7
8 DATASET_PATH = "data.json"
9
10 def load_data(dataset_path):
11     with open(dataset_path, "r") as fp:
12         data = json.load(fp)
13
14     # convert lists into numpy arrays
15     inputs = np.array(data["mfcc"])
16     targets = np.array(data["labels"])
17
18     return inputs, targets
19
20 def plot_history(history):
21     fig, axs = plt.subplots(2)
22
23     # accuracy subplot
24     axs[0].plot(history.history["accuracy"], label="train accuracy")
25     axs[0].plot(history.history["val_accuracy"], label="test accuracy")
26     axs[0].set_ylabel("Accuracy")
27     axs[0].legend(loc="lower right")
28     axs[0].set_title("Accuracy evaluation")
29
30     # error subplot
31     axs[1].plot(history.history["loss"], label="train error")
32     axs[1].plot(history.history["val_loss"], label="test error")
33     axs[1].set_ylabel("Error")
34     axs[1].set_xlabel("Epoch")
35     axs[1].legend(loc="lower right")
36     axs[1].set_title("Error val")
37
38     plt.show()
39
40 def prep_data(inputs, targets):
41     inputs_train, inputs_test, targets_train, targets_test \
42         = train_test_split(inputs, targets, test_size=0.45)
43     inputs_val, inputs_test, targets_val, targets_test \
44         = train_test_split(inputs_test, targets_test, test_size=0.45)
45
46     return inputs_train, inputs_val, inputs_test, \
47         targets_train, targets_val, targets_test
48
49 if __name__ == "__main__":

```



```

50 # load data
51 inputs, targets = load_data(DATASET_PATH)
52
53 # split the data into train and test sets
54 inputs_train, inputs_val, inputs_test, targets_train, targets_val, targets_test
55 = prep_data(inputs, targets)
56
57 # build the network achitecture
58 model = keras.Sequential([
59     # input layer
60     keras.layers.Flatten(input_shape=(inputs.shape[1], inputs.shape[2])),
61
62     # 1st hidden layer
63     keras.layers.Dense(512, activation="relu", kernel_regularizer=keras.
64     regularizers.l2(0.001)),
65     keras.layers.Dropout(0.3),
66
67     # 2nd hidden layer
68     keras.layers.Dense(256, activation="relu", kernel_regularizer=keras.
69     regularizers.l2(0.001)),
70     keras.layers.Dropout(0.3),
71
72     # 3rd hidden layer
73     keras.layers.Dense(64, activation="relu", kernel_regularizer=keras.
74     regularizers.l2(0.001)),
75     keras.layers.Dropout(0.3),
76
77     # output layer
78     keras.layers.Dense(10, activation="softmax")
79 ])
80
81 # compile network
82 optimizer = keras.optimizers.legacy.Adam(learning_rate=0.0001)
83 model.compile(optimizer,
84               loss="sparse_categorical_crossentropy",
85               metrics=["accuracy"])
86 model.summary()
87
88 # train the network
89 history = model.fit(inputs_train, targets_train,
90                    validation_data=(inputs_val, targets_val),
91                    batch_size=32, epochs=80)
92
93 plot_history(history)
94
95 predictions = model.predict(inputs_test)
96 predicted_labels = np.argmax(predictions, axis=1)
97
98 # calculate metrics
99 accuracy = accuracy_score(targets_test, predicted_labels)
100 precision = precision_score(targets_test, predicted_labels, average="weighted")

```

```
98     recall = recall_score(targets_test, predicted_labels, average="weighted")
99     f1 = f1_score(targets_test, predicted_labels, average="weighted")
100
101     print(f"Accuracy: {accuracy}")
102     print(f"Precision: {precision}")
103     print(f"Recall: {recall}")
104     print(f"F1 Score: {f1}")
```

Κώδικας 8: CNN μοντέλο

```

1 import json
2 import numpy as np
3 import keras
4 from sklearn.model_selection import train_test_split
5 from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
6 import matplotlib.pyplot as plt
7
8 DATASET_PATH = "data.json"
9
10 def load_data(dataset_path):
11     with open(dataset_path, "r") as fp:
12         data = json.load(fp)
13         # convert lists into numpy arrays
14         inputs = np.array(data["mfcc"])
15         targets = np.array(data["labels"])
16
17         return inputs, targets
18
19 def plot_history(history):
20     fig, axs = plt.subplots(2)
21
22     # accuracy subplot
23     axs[0].plot(history.history["accuracy"], label="train accuracy")
24     axs[0].plot(history.history["val_accuracy"], label="test accuracy")
25     axs[0].set_ylabel("Accuracy")
26     axs[0].legend(loc="lower right")
27     axs[0].set_title("Accuracy evaluation")
28
29     # error subplot
30     axs[1].plot(history.history["loss"], label="train error")
31     axs[1].plot(history.history["val_loss"], label="test error")
32     axs[1].set_ylabel("Error")
33     axs[1].set_xlabel("Epoch")
34     axs[1].legend(loc="lower right")
35     axs[1].set_title("Error val")
36
37     plt.show()
38
39 def print_metrics(inputs_test):
40     predictions = model.predict(inputs_test)
41     predicted_labels = np.argmax(predictions, axis=1)
42
43     # calculate metrics
44     accuracy = accuracy_score(targets_test, predicted_labels)
45     precision = precision_score(targets_test, predicted_labels, average="weighted")
46     recall = recall_score(targets_test, predicted_labels, average="weighted")
47     f1 = f1_score(targets_test, predicted_labels, average="weighted")
48
49     print(f"Accuracy: {accuracy}")
50     print(f"Precision: {precision}")
51     print(f"Recall: {recall}")

```

```

52     print(f"F1 Score: {f1}")
53
54 def prep_data(inputs, targets):
55     inputs_train, inputs_test, targets_train, targets_test \
56         = train_test_split(inputs, targets, test_size=0.45)
57     inputs_test, inputs_val, targets_test, targets_val \
58         = train_test_split(inputs_test, targets_test, test_size=0.45)
59     inputs_train = inputs_train[..., np.newaxis]
60     inputs_val = inputs_val[..., np.newaxis]
61     inputs_test = inputs_test[..., np.newaxis]
62
63     return inputs_train, inputs_val, inputs_test, \
64           targets_train, targets_val, targets_test
65
66 def predict(model, input, expected_output):
67
68     input = input[np.newaxis, ...] # array shape (1, 130, 13, 1)
69
70     # perform prediction
71     prediction = model.predict(input)
72
73     # get index with max value
74     predicted_index = np.argmax(prediction, axis=1)
75
76     print("Target: {}, Predicted label: {}".format(expected_output, predicted_index)
77         )
78
79 def build_model(input_shape):
80     # create model
81     model = keras.Sequential()
82
83     # first conv layer
84     model.add(keras.layers.Conv2D(32, (3,3), activation='relu', input_shape=
85         input_shape))
86     model.add(keras.layers.MaxPool2D((3,3), strides=(2, 2), padding='same'))
87     model.add(keras.layers.BatchNormalization())
88
89     # second conv layer
90     model.add(keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=
91         input_shape))
92     model.add(keras.layers.MaxPool2D((3, 3), strides=(2, 2), padding='same'))
93     model.add(keras.layers.BatchNormalization())
94
95     # third conv layer
96     model.add(keras.layers.Conv2D(32, (2, 2), activation='relu', input_shape=
97         input_shape))
98     model.add(keras.layers.MaxPool2D((2, 2), strides=(2, 2), padding='same'))
99     model.add(keras.layers.BatchNormalization())
100
101     # flatten output of conv layer and feed it into dense layer
102     model.add(keras.layers.Flatten())
103     model.add(keras.layers.Dense(64, activation='relu'))

```

```

100     model.add(keras.layers.Dropout(0.3))
101
102     # output layer that uses softmax
103     model.add(keras.layers.Dense(10, activation='softmax'))
104
105     return model
106
107 if __name__=="__main__":
108     inputs, targets = load_data(DATASET_PATH)
109     inputs_train, inputs_val, inputs_test, targets_train, targets_val, targets_test
110     = prep_data(inputs, targets)
111
112     # Build CNN
113     input_shape = (inputs_train.shape[1], inputs_train.shape[2], inputs_train.shape
114     [3])
115     model = build_model(input_shape)
116
117     # Compile the CNN
118     optimizer = keras.optimizers.legacy.Adam(learning_rate=0.0001)
119     model.compile(optimizer=optimizer,
120     loss="sparse_categorical_crossentropy",
121     metrics=['accuracy'])
122
123     # Train Model
124     history = model.fit(inputs_train, targets_train,
125     validation_data=(inputs_val, targets_val),
126     batch_size=32, epochs=50)
127     model.summary()
128     error, accuracy = model.evaluate(inputs_test, targets_test, verbose=1)
129     print(f"Accuracy on test = {accuracy}")
130     plot_history(history)
131     print_metrics(inputs_test)
132
133     # # pick a sample to predict from the test set
134     # X_to_predict = inputs_test[100]
135     # y_to_predict = targets_test[100]
136     #
137     # # predict sample
138     # predict(model, X_to_predict, y_to_predict)
139
140     # model.save('mymodel.keras')

```

Κώδικας 9: LSTM μοντέλο

```

1 import json
2 import numpy as np
3 import keras
4 from sklearn.model_selection import train_test_split
5 from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
6 import matplotlib.pyplot as plt
7
8 DATASET_PATH = "data.json"
9
10 def load_data(dataset_path):
11     with open(dataset_path, "r") as fp:
12         data = json.load(fp)
13         # convert lists into numpy arrays
14         inputs = np.array(data["mfcc"])
15         targets = np.array(data["labels"])
16
17         return inputs, targets
18
19 def plot_history(history):
20     fig, axs = plt.subplots(2)
21
22     # accuracy subplot
23     axs[0].plot(history.history["accuracy"], label="train accuracy")
24     axs[0].plot(history.history["val_accuracy"], label="test accuracy")
25     axs[0].set_ylabel("Accuracy")
26     axs[0].legend(loc="lower right")
27     axs[0].set_title("Accuracy evaluation")
28
29     # error subplot
30     axs[1].plot(history.history["loss"], label="train error")
31     axs[1].plot(history.history["val_loss"], label="test error")
32     axs[1].set_ylabel("Error")
33     axs[1].set_xlabel("Epoch")
34     axs[1].legend(loc="lower right")
35     axs[1].set_title("Error val")
36
37     plt.show()
38
39 def print_metrics(inputs_test):
40     predictions = model.predict(inputs_test)
41     predicted_labels = np.argmax(predictions, axis=1)
42
43     # calculate metrics
44     accuracy = accuracy_score(targets_test, predicted_labels)
45     precision = precision_score(targets_test, predicted_labels, average="weighted")
46     recall = recall_score(targets_test, predicted_labels, average="weighted")
47     f1 = f1_score(targets_test, predicted_labels, average="weighted")
48
49     print(f"Accuracy: {accuracy}")
50     print(f"Precision: {precision}")
51     print(f"Recall: {recall}")

```

```

52     print(f"F1 Score: {f1}")
53
54 def prep_data(inputs, targets):
55     inputs_train, inputs_test, targets_train, targets_test \
56         = train_test_split(inputs, targets, test_size=0.45)
57     inputs_test, inputs_val, targets_test, targets_val \
58         = train_test_split(inputs_test, targets_test, test_size=0.45)
59
60     return inputs_train, inputs_val, inputs_test, \
61           targets_train, targets_val, targets_test
62
63 def predict(model, input, expected_output):
64
65     input = input[np.newaxis, ...] # array shape (1, 130, 13, 1)
66
67     # perform prediction
68     prediction = model.predict(input)
69
70     # get index with max value
71     predicted_index = np.argmax(prediction, axis=1)
72
73     print("Target: {}, Predicted label: {}".format(expected_output, predicted_index)
74           )
75
76 def build_model(input_shape):
77     # create model
78     model = keras.Sequential()
79
80     # 2 LSTM layers
81     model.add(keras.layers.LSTM(64, input_shape=input_shape, return_sequences=True))
82     model.add(keras.layers.LSTM(64))
83
84     # dense layer
85     model.add(keras.layers.Dense(64, activation='relu'))
86     model.add(keras.layers.Dropout(0.3))
87
88     # output layer
89     model.add(keras.layers.Dense(10, activation='softmax'))
90
91     return model
92
93 if __name__=="__main__":
94     inputs, targets = load_data(DATASET_PATH)
95     inputs_train, inputs_val, inputs_test, targets_train, targets_val, targets_test
96     = prep_data(inputs, targets)
97
98     # Build CNN
99     input_shape = (inputs_train.shape[1], inputs_train.shape[2])
100    model = build_model(input_shape)
101
102    # Compile the CNN
103    optimizer = keras.optimizers.legacy.Adam(learning_rate=0.0001)

```

```

102 model.compile(optimizer=optimizer,
103               loss="sparse_categorical_crossentropy",
104               metrics=['accuracy'])
105
106 model.summary()
107 # Train Model
108 history = model.fit(inputs_train, targets_train,
109                   validation_data=(inputs_val, targets_val),
110                   batch_size=32, epochs=50)
111
112 error, accuracy = model.evaluate(inputs_test, targets_test, verbose=2)
113 print(f"Accuracy on test = {accuracy}")
114 plot_history(history)
115 print_metrics(inputs_test)

```

Κώδικας 10: Διεπαφή Κονσόλας

```

1 import json
2 import numpy as np
3 import keras
4 import librosa, librosa.feature
5 from sklearn.model_selection import train_test_split
6 import matplotlib.pyplot as plt
7 from scipy.spatial.distance import cosine
8
9 DATASET_PATH = "MGC/data.json"
10 DATASET_PATH_RS = "Recommender Systems/recommendation_data.json"
11 SAMPLES_PER_TRACK = 22050 * 30
12 num_samples_per_segment = int(SAMPLES_PER_TRACK / 10)
13
14
15 def load_data(dataset_path):
16     with open(dataset_path, "r") as fp:
17         data = json.load(fp)
18         inputs = np.array(data["mfcc"])
19         targets = np.array(data["labels"])
20         return inputs, targets
21
22
23 def plot_results(prediction):
24     x_axis = ["pop", "metal", "disco", "blues", "reggae", "classical", "rock", "
25             hiphop", "country", "jazz"]
26     y_axis = []
27     for p in prediction:
28         y_axis.append(p)
29     plt.figure(figsize=(10, 6))
30     plt.bar(x_axis, y_axis)
31     plt.xlabel('Genres')
32     plt.ylabel('Score')
33     plt.show()
34
35 def predict(model, song, mfcc_rs):

```



```

36 sum_preds = np.zeros(10)
37 for s in song:
38     s = s[np.newaxis, ...]
39     prediction = model.predict(s)
40     preds = np.asarray(prediction[0])
41     sum_preds = np.add(sum_preds, preds)
42 plot_results(sum_preds)
43 genres = ["pop", "metal", "disco", "blues", "reggae", "classical", "rock", "
hiphop", "country", "jazz"]
44 predicted_genre = np.argmax(sum_preds) # Genre of Music that was mostly
predicted
45 index = find_most_similar_song(mfcc_rs, list_of_songs, genres[predicted_genre])
46 for k, i in enumerate(index):
47     print(f'Top {k + 1} = {song_metadata["name"][i]}, path {song_metadata["path
"] [i]}')
48
49 def find_most_similar_song(song, list_of_songs, predicted_genre):
50     cosine_similarity_score = []
51     i = 0
52     for s in list_of_songs:
53         if predicted_genre in song_metadata["name"][i]:
54             cosine_similarity_score.append(1 - cosine(song.flatten(), s.flatten()))
55         else:
56             cosine_similarity_score.append(-1)
57         i = i + 1
58     top_indices = sorted(enumerate(cosine_similarity_score), key=lambda x: x[1],
reverse=True)[:5]
59     top_indices_only = [index for index, _ in top_indices]
60     return top_indices_only
61
62 def load_song(song_path):
63     signal, sr = librosa.load(song_path, sr=22050)
64     mfcc_table = []
65     for s in range(10):
66         start_sample = num_samples_per_segment * s
67         finish_sample = start_sample + num_samples_per_segment
68         mfcc = librosa.feature.mfcc(y=signal[start_sample:finish_sample], n_mfcc=13,
n_fft=2048, hop_length=512).T
69         mfcc = mfcc[... , np.newaxis]
70         mfcc_table.append(mfcc)
71
72     # For RS
73     if len(signal) > 1984512:
74         mfcc_rs = librosa.feature.mfcc(y=signal[882000:1543504], sr=sr, n_mfcc=40,
n_fft=2048, hop_length=512).T
75     else:
76         mfcc_rs = librosa.feature.mfcc(y=signal[:661504], sr=sr, n_mfcc=40, n_fft
=2048, hop_length=512).T
77
78     return mfcc_table, mfcc_rs
79
80

```

```
81
82 if __name__=="__main__":
83     # Load the Model
84     model = keras.models.load_model('MGC/mymodel.keras')
85     input_song = input("Give path of song:\n")
86
87     with open(DATASET_PATH_RS,"r") as fp:
88         data = json.load(fp)
89         list_of_songs = np.array(data["mfcc"])
90         song_metadata = {"name": np.array(data["name"]), "path": np.array(data["path"])}
91
92     while input_song != "":
93         # Load song
94         song, mfcc_rs = load_song(input_song)
95         predict(model, song, mfcc_rs)
96         input_song = input("Give path of song:\n")
```

Κώδικας 11: Recommendation Engine

```

1 import json
2 import numpy as np
3 import librosa, librosa.feature
4 import math
5 from scipy.spatial.distance import cosine
6
7 DATASET_PATH = "recommendation_data.json"
8 SAMPLES_PER_TRACK = 22050 * 30
9 num_samples_per_segment = int(SAMPLES_PER_TRACK / 10)
10
11
12 def load_data(dataset_path):
13     # convert lists into numpy arrays
14     inputs = np.array(data["mfcc"])
15     return inputs
16
17
18 def load_song(song_path):
19     signal, sr = librosa.load(song_path, sr=22050)
20     if len(signal) > 1984512:
21         mfcc = librosa.feature.mfcc(y=signal[882000:1543504], sr=sr, n_mfcc=40,
22             n_fft=2048, hop_length=512).T
23     else:
24         mfcc = librosa.feature.mfcc(y=signal[:661504], sr=sr, n_mfcc=40, n_fft=2048,
25             hop_length=512).T
26     return mfcc
27
28 def find_most_similar_song(song, list_of_songs):
29     cosine_similarity_score = [1 - cosine(song.flatten(), s.flatten()) for s in
30         list_of_songs]
31     top_indices = sorted(enumerate(cosine_similarity_score), key=lambda x: x[1],
32         reverse=True)[:5]
33     top_indices_only = [index for index, _ in top_indices]
34
35     # With Euclidean
36     EMD_similarity_score = [math.dist(song.flatten(), s.flatten()) for s in
37         list_of_songs]
38     EMD_top_indices_only = sorted(range(len(EMD_similarity_score)), key=lambda sub:
39         EMD_similarity_score[sub])[:5]
40
41     return top_indices_only, EMD_top_indices_only
42
43
44 if __name__ == "__main__":
45     with open(DATASET_PATH, "r") as fp:
46         data = json.load(fp)
47
48     list_of_songs = np.array(data["mfcc"])
49     song_metadata = {"name": np.array(data["name"]), "path": np.array(data["path"])}

```

```
46 input_song = input("Input path:\n")
47 while input_song != "":
48     # Load Song
49     song = load_song(input_song)
50     index, EMD_index = find_most_similar_song(song, list_of_songs)
51     print("Results with Cosine Similarity")
52     for k, i in enumerate(index):
53         print(f'Top {k+1} = {song_metadata["name"][i]}, path {song_metadata["path"][i]}')
54     print("Results with Euclidean Distance")
55     for k, i in enumerate(EMD_index):
56         print(f'Top with EMD{k+1} = {song_metadata["name"][i]}, path {song_metadata["path"][i]}')
57     input_song = input("Input path:\n")
```

Κώδικας 12: Recommendation Engine με διάφορα Features

```

1 import librosa, librosa.display, librosa.feature
2 import numpy as np
3 import pandas as pd
4 from scipy.spatial.distance import cosine
5 import math
6
7 CSV_PATH = "/Users/conspd/School/Thesis/Data/features_30_sec.csv"
8 sr = 22050
9 hop_length = 512
10 n_fft = 2048
11
12 def get_song_list():
13     with open(CSV_PATH) as f:
14         df = pd.read_csv(CSV_PATH)
15         df.drop(['length', 'label', 'harmony_mean', 'harmony_var', 'perceptr_mean', '
perceptr_var'], axis=1, inplace=True)
16         data = np.array(df.values.tolist())
17     return data
18
19 def extract_features(y, start, finish):
20     # Chroma
21     chroma_stft = librosa.feature.chroma_stft(y=y, sr=sr)
22     chroma_stft_mean = np.mean(chroma_stft)
23     chroma_stft_var = np.var(chroma_stft)
24
25     # RMS
26     rms = librosa.feature.rms(y=y)
27     rms_mean = np.mean(rms)
28     rms_var = np.var(rms)
29
30     # Spectral Centroid
31     spectral_centroid = librosa.feature.spectral_centroid(y=y, sr=sr)
32     spectral_centroid_mean = np.mean(spectral_centroid)
33     spectral_centroid_var = np.var(spectral_centroid)
34
35     # Spectral Bandwidth
36     spectral_bandwidth = librosa.feature.spectral_bandwidth(y=y, sr=sr)
37     spectral_bandwidth_mean = np.mean(spectral_bandwidth)
38     spectral_bandwidth_var = np.var(spectral_bandwidth)
39
40     # Rolloff
41     rolloff = librosa.feature.spectral_rolloff(y=y, sr=sr)
42     rolloff_mean = np.mean(rolloff)
43     rolloff_var = np.var(rolloff)
44
45     # Zero Crossing Rate
46     zcr = librosa.feature.zero_crossing_rate(y)
47     zcr_mean = np.mean(zcr)
48     zcr_var = np.var(zcr)
49
50     # Tempo

```

```

51 tempo = librosa.feature.tempo(y=y, sr=sr)[0]
52
53 # MFCC
54 mfcc = librosa.feature.mfcc(y=y[start:finish], sr=sr)
55 mfcc_mean = [np.mean(m) for m in mfcc]
56 mfcc_var = [np.var(m) for m in mfcc]
57
58 features = [chroma_stft_mean, chroma_stft_var,
59             rms_mean, rms_var,
60             spectral_centroid_mean, spectral_centroid_var,
61             spectral_bandwidth_mean, spectral_bandwidth_var,
62             rolloff_mean, rolloff_var,
63             zcr_mean, zcr_var,
64             tempo]
65 for m, v in zip(mfcc_mean, mfcc_var):
66     features.append(m)
67     features.append(v)
68
69 return features
70
71 def load_song(song_path):
72     signal, sr = librosa.load(song_path, sr=22050)
73     if len(signal) > 1984512:
74         features = extract_features(signal, 882000, 1543504)
75     else:
76         features = extract_features(signal, 0, 661504)
77     return features
78
79
80 def find_most_similar_song(song, song_list):
81     cosine_similarity_score = [1 - cosine(song, np.array(s[1:]).astype(float)) for s
82                               in song_list]
83     top_indices = sorted(enumerate(cosine_similarity_score), key=lambda x: x[1],
84                          reverse=True)[:5]
85     top_indices_only = [index for index, _ in top_indices]
86
87     # With Euclidean
88     EMD_similarity_score = [math.dist(song, np.array(s[1:]).astype(float)) for s in
89                             song_list]
90     EMD_top_indices_only = sorted(range(len(EMD_similarity_score)), key=lambda sub:
91                                   EMD_similarity_score[sub])[:5]
92
93     return top_indices_only, EMD_top_indices_only
94
95 def print_results(cosine_top, emd_top, song_list):
96     for k, i in enumerate(cosine_top):
97         print(f'Top {k + 1} = {song_list[i][0]}')
98     print("Results with Euclidean Distance")
99     for k, i in enumerate(emd_top):
100        print(f'Top EMD{k + 1} = {song_list[i][0]}')
101
102 def run_app():

```

```
99     song_list = get_song_list()
100     while True:
101         song_path = input("Input Path:\n")
102         song = load_song(song_path)
103         cosine_top, emd_top = find_most_similar_song(song, song_list)
104         print_results(cosine_top, emd_top, song_list)
105
106 if __name__ == "__main__":
107     run_app()
```

Αναφορές

- [1] Hareesh Bahuleyan. Music genre classification using machine learning techniques. *arXiv preprint arXiv:1804.01149*, 2018.
- [2] Wangduk Seo, Sung-Hyun Cho, Paweł Teisseyre, and Jaesung Lee. A short survey and comparison of cnn-based music genre classification using multiple spectral features. *IEEE Access*, 2023.
- [3] Peter Knees and Markus Schedl. A survey of music similarity and recommendation from music context data. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 10(1):1–21, 2013.
- [4] Peter J Rentfrow and Samuel D Gosling. The do re mi's of everyday life: the structure and personality correlates of music preferences. *Journal of personality and social psychology*, 84(6):1236, 2003.
- [5] Daniel Walzer. Independent music production: How individuality, technology, and creative entrepreneurship influence contemporary music industry practices. *Creative Industries Journal*, 11 2016.
- [6] Are dominant platforms good for consumers. *Economic Inquiry*, 59(3):1364–1377, 2021.
- [7] Antje Cockrill, Margaret Sullivan, and Heather L Norbury. Music consumption: Lifestyle choice or addiction. *Journal of Retailing and Consumer Services*, 18(2):160–166, 2011.
- [8] Dillon Ranwala. The evolution of music and ai technology, 2020.
- [9] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.
- [10] Andrea Zugarini, Stefano Melacci, and Marco Maggini. Neural poetry: Learning to generate poems using syllables. 8 2019.
- [11] Oren Sharon & Arielle Spivac. What's going on in the world of ai and music today, Mar 2023.
- [12] Andrea Agostinelli, Timo I. Denk, Zalán Borsos, Jesse Engel, Mauro Verzetti, Antoine Caillon, Qingqing Huang, Aren Jansen, Adam Roberts, Marco Tagliasacchi, Matt Sharifi, Neil Zeghidour, and Christian Frank. Musiclm: Generating music from text, 2023.
- [13] Jade Copet, Felix Kreuk, Itai Gat, Tal Remez, David Kant, Gabriel Synnaeve, Yossi Adi, and Alexandre Défossez. Simple and controllable music generation, 2023.
- [14] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music, 2020.
- [15] Markus Schedl, Emilia Gómez, and Julián Urbano. *Music Information Retrieval: Recent Developments and Applications*. 2014.
- [16] Emiru Tsunoo, George Tzanetakis, Nobutaka Ono, and Shigeki Sagayama. Beyond timbral statistics: Improving music classification using percussive patterns and bass lines. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(4):1003–1014, 2010.
- [17] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, 2002.
- [18] Michael I Mandel and Daniel PW Ellis. Song-level features and support vector machines for music classification. 2005.
- [19] Janet Marques and Pedro J Moreno. A study of musical instrument classification using gaussian mixture models and support vector machines. *Cambridge Research Laboratory Technical Report Series CRL*, 4:143, 1999.
- [20] Maros Jakubec and Michal Chmulik. Automatic music genre recognition for in-car infotainment. *Transportation Research Procedia*, 40:1364–1371, 2019.
- [21] Yinhui Yi, Xiaohui Zhu, Yong Yue, and Wei Wang. Music genre classification with lstm based on time and frequency domain features. pages 678–682, 04 2021.
- [22] Dipjyoti Bisharad and Rabul Hussain Laskar. Music genre recognition using residual neural networks. *TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON)*, pages 2063–2068, 2019.

- [23] Fika Ulfa Widowati, Fitriyanto Santi Nugroho, and Guruh Fajar Shidik. Classification of music moods based on cnn. In *2018 International Seminar on Application for Technology of Information and Communication*, pages 318–321, 2018.
- [24] Nantalira Niar Wijaya, Ahmad Rofiqul Muslikh, et al. Music-genre classification using bidirectional long short-term memory and mel-frequency cepstral coefficients. *Journal of Computing Theories and Applications*, 2(1):13–26, 2024.
- [25] Fady Medhat, David Chesmore, and John Robinson. Masked conditional neural networks for sound classification. *Applied Soft Computing*, 90:106073, 2020.
- [26] B Jaishankar, Raghunathan Anitha, Finney Daniel Shadrach, M Sivarathinabala, and V Balamurugan. Music genre classification using african buffalo optimization. *Computer Systems Science & Engineering*, 44(2), 2023.
- [27] Álvaro Lozano Murciego, Diego M Jiménez-Bravo, Adrián Valera Román, Juan F De Paz Santana, and María N Moreno-García. Context-aware recommender systems in the music domain: A systematic literature review. *Electronics*, 10(13):1555, 2021.
- [28] Marius Kaminskis and Francesco Ricci. Contextual music information retrieval and recommendation: State of the art and challenges. *Computer Science Review*, 6(2-3):89–119, 2012.
- [29] Ju-Chiang Wang, Hung-Shin Lee, Hsin-min Wang, and Shyh-Kang Jeng. Learning the similarity of audio music in bag-of-frames representation from tagged music data. pages 85–90, 01 2011.
- [30] Dingding Wang, Tao Li, and Mitsunori Ogihara. Are tags better than audio? the effect of joint use of tags and audio content features for artistic style clustering. pages 57–62, 01 2010.
- [31] Panagiotis Symeonidis, Maria Ruxanda, Alexandros Nanopoulos, and Yannis Manolopoulos. Ternary semantic analysis of social tags for personalized music recommendation. pages 219–224, 01 2008.
- [32] Douglas Eck, Paul Lamere, Thierry Bertin-Mahieux, and Stephen Green. Automatic generation of social tags for music recommendation. 01 2007.
- [33] Peter Knees and Markus Schedl. A survey of music similarity and recommendation from music context data. *ACM Trans. Multimedia Comput. Commun. Appl.*, 10(1), 2013.
- [34] Aaron van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.
- [35] Michael A Casey, Remco Veltkamp, Masataka Goto, Marc Leman, Christophe Rhodes, and Malcolm Slaney. Content-based music information retrieval: Current directions and future challenges. *Proceedings of the IEEE*, 96(4):668–696, 2008.
- [36] Ning Hu and Roger B Dannenberg. A comparison of melodic database retrieval techniques using sung queries. In *Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries*, pages 301–307, 2002.
- [37] Eleanor Selfridge-Field. Conceptual and representational issues in melodic comparison. *Computing in Musicology*, 11:3–64, 1998.
- [38] Beth Logan and Ariel Salomon. A music similarity function based on signal analysis. *International Conference on Multimedia and Expo*, 06 2001.
- [39] Tao Li and Mitsunori Ogihara. Content-based music similarity search and emotion detection. In *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 5, pages V–705. IEEE, 2004.
- [40] Tetsuro Kitahara, Yusuke Tsuchihashi, and Haruhiro Katayose. Music genre classification and similarity calculation using bass-line features. In *2008 Tenth IEEE International Symposium on Multimedia*, pages 574–579. IEEE, 2008.
- [41] George Tzanetakis. Marsyas submissions to mirex 2012. *Proceedings of the Music Information Retrieval Evaluation Exchange*. [Cited in page 102.], 2012.
- [42] Tim Pohle, Dominik Schnitzer, Markus Schedl, Peter Knees, and Gerhard Widmer. On rhythm and general music similarity. In *ISMIR*, pages 525–530, 2009.
- [43] Keunwoo Choi, György Fazekas, Kyunghyun Cho, and Mark Sandler. A tutorial on deep learning for music information retrieval. 9 2017.

- [44] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.
- [45] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [46] Rémi Mignot and Geoffroy Peeters. An analysis of the effect of data augmentation methods: experiments for a musical genre classification task. *Transactions of the International Society for Music Information Retrieval*, 2(1), 2019.
- [47] J. Stephen Downie. Music information retrieval. *Annual Review of Information Science and Technology*, 37:295–340, 2003.
- [48] Sep 2023.
- [49] Akram Roshdi and Akram Roohparvar. Review: Information retrieval techniques and applications, 2015.
- [50] Mark Sanderson and W Bruce Croft. The history of information retrieval research, 2012.
- [51] Beauty & AI. History of ai art from 1960s to 2023, 2023.
- [52] Robin Burke. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12:331–370, 2002.
- [53] Dmitry Bogdanov, Herrera Boyer, et al. How much metadata do we need in music recommendation?: A subjective evaluation using preference sets. In *Klapuri A, Leider C, editors. 12th International Society for Music Information Retrieval Conference (ISMIR 2011); 2011 Oct 24-28; Miami, USA. Montréal: ISMIR; 2011. p. 97-102*. International Society for Music Information Retrieval (ISMIR), 2011.
- [54] Malcolm Slaney. Web-scale multimedia analysis: Does content matter? *IEEE MultiMedia*, 18(2):12–15, 2011.
- [55] LI Luyang. A survey of content-based music similarity.