ΥΣ19 Artificial Intelligence II

Τρίτη Εργασία.

Ονοματεπώνυμο: Απόστολος Καρβέλας

Α.Μ.: 1115201800312

Πρώτο υποερώτημα

ΟΔΗΓΙΕΣ ΧΡΗΣΗΣ

Για να γίνει εκτέλεση των πειραμάτων πρέπει πρώτα αν έχουν τρέξει τα αρχικά κελιά και συγκεκριμένα τα imports, preprocessing & print functions, download GloVe και το read files. Στην συνέχεια, υπάρχουν ξεχωριστά κελιά για κάθε πείραμα με τον αντίστοιχο τίτλο του και στο τέλος υπάρχει το καλύτερο μοντέλο, δηλαδή εκείνο με τα βέλτιστα αποτελέσματα. Για να τρέξει ένα κελί με διαφορετικά δεδομένα πρέπει πρώτα να αλλάξει το path των αρχείων που βρίσκεται στην αρχή κάθε κελιού.

ΕΙΣΑΓΩΓΗ

Στην εργασία αυτή θα ασχοληθούμε με τον πειραματισμό των RNN δικτύων και τους τύπους κελιών LSTM και GRU. Συγκεκριμένα σταδιακά θα ερευνήσουμε:

- Vanilla RNN δίκτυα, διάφορων μεγεθών
- Multilayer RNN, για διαφορικά επίπεδα και μεγέθη
- Dropout
- Gradient Clipping
- Vanilla LSTM
- Multilayer LSTM
- Bidirectional
- GRU
- Skip connections

Και θα προσπαθήσουμε να καταλήξουμε σε ένα καλό μοντέλο για τα δεδομένα αυτά. Τέλος θα συγκρίνουμε το μοντέλο με αυτά των προηγούμενων εργασιών.

Recurrent neural network

Αρχικά θα παρατηρήσουμε την συμπεριφορά των RNN χρησιμοποιώντας ένα απλό νευρωνικό δίκτυο. Για τα πειράματα θα χρησιμοποιήσουμε τις εξής τιμές, τις οποίες από προηγούμενες εργασίες έχουμε καταλήξει ότι επηρεάζουν θετικά τα μοντέλα και βοηθάνε στην επεξεργασία και τον πειραματισμό τους :

batch size: 64

Optimizer: Adam

Learning rate: 10 e-5

Number of Epochs: 50

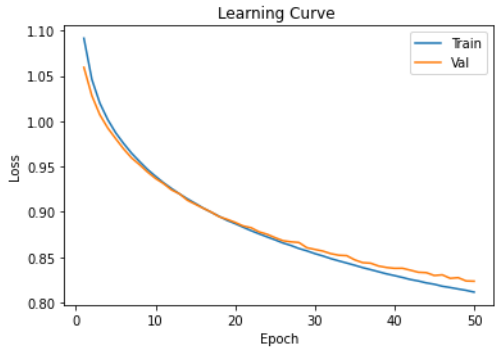
Loss function: Cross-entropy loss

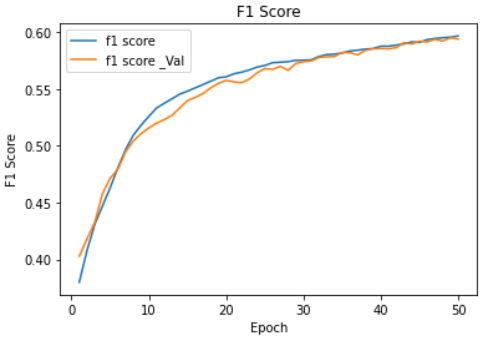
Με preprocessing και την χρήση GloVe 300 διαστάσεων.

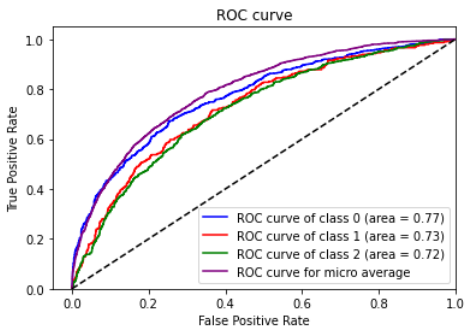
Για την κατανόηση των RNN θα ξεκινήσουμε με 1 layer και hidden size 32.

Training loss = 0.8116105 Validation loss = 0.8236163

Training F1_score = 0.5966675 Validation F1_score = 0.5939441







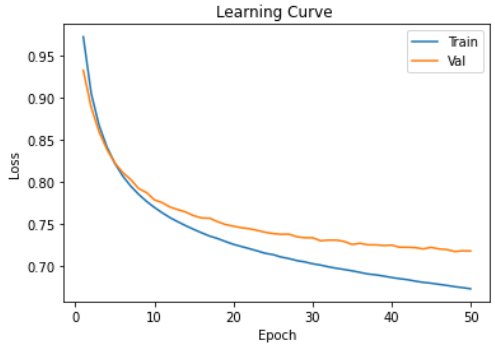
Παρατηρούμε ότι το μοντέλο αυτό είναι πολύ απλοϊκό και έχει σαν αποτέλεσμα την δυσκολία μάθησης των δεδομένων. Το γεγονός αυτό φαίνεται καθαρά στην τιμή του f1 score και των training όπως και του validation αφού έχουν μικρή τιμή κοντά στο 0.6. Επίσης από τον loss-epochs γράφο βλέπουμε ότι δεν προκαλείται overfitting αφού οι τιμές είναι παρόμοιες αλλά αντίθετα underfitting αφού οι τιμές είναι υψηλές. Οι συμπεριφορά αυτή

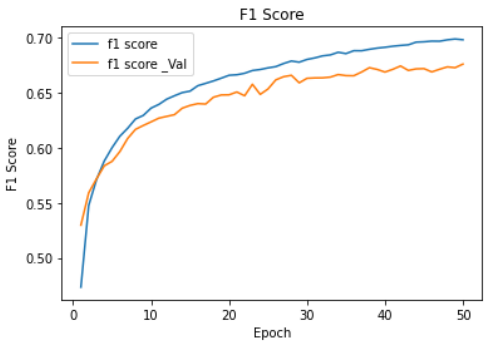
είναι λογική διότι το μοντέλο είναι πολύ απλό και δεν μπορεί να μάθει πολλές τιμές των δεδομένων μας.

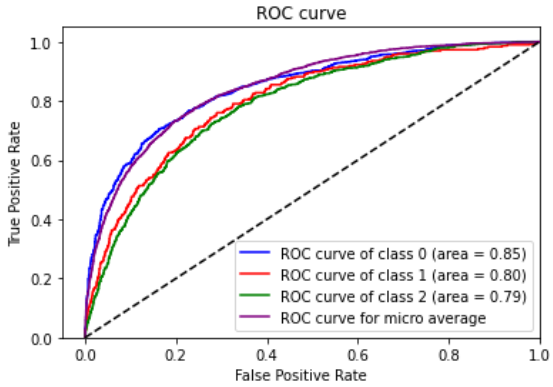
Για τον λόγω αυτόν θα προσπαθήσουμε να αυξήσουμε τον αριθμό του hidden size ώστε να μεγαλώσει το μέγεθος του πίνακα με features για το hidden state. Με αυτόν τον τρόπο το μοντέλο θα γίνει πιο περίπλοκο και πρέπει να έχει σαν αποτέλεσμα την αύξηση των σωστών αποτελεσμάτων. Οπότε θα αλλάξουμε το hidden size σε 256.

Training loss = 0.6735584 Validation loss = 0.7185402

Training F1_score = 0.6981008 Validation F1_score = 0.6759926





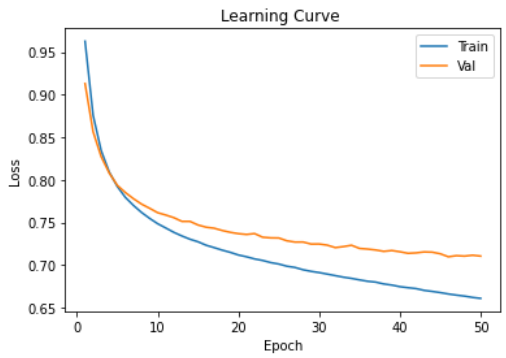


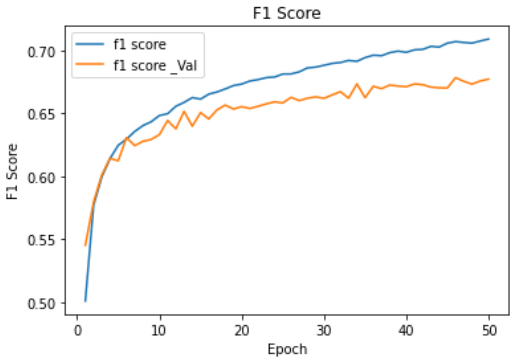
Από τα αποτελέσματα βλέπουμε ότι ενώ το μοντέλο παρουσιάζει λίγο overfitting, τα f1 scores είναι πολύ καλύτερα από το προηγούμενο παράδειγμα οπότε η υπόθεση ήταν σωστή. Το overfitting μπορεί να εμφανίζεται από την έλλειψη διάφορων τεχνικών κανονικοποίησης, όπως η χρήση dropout μεταξύ των layers.

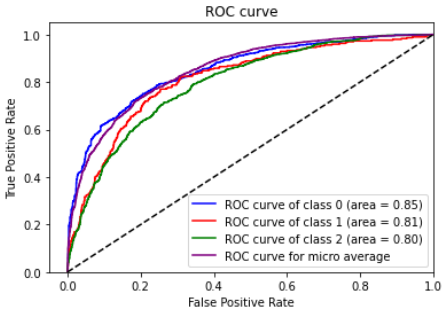
Για να σιγουρέψουμε την θεωρία θα δοκιμάσουμε να αυξήσουμε πάλι το hidden size σε 512 και να παρατηρήσουμε τις διαφορές του.

Training loss = 0.6608657 Validation loss = 0.7104702

Training F1_score = 0.7092211 Validation F1_score = 0.6773614







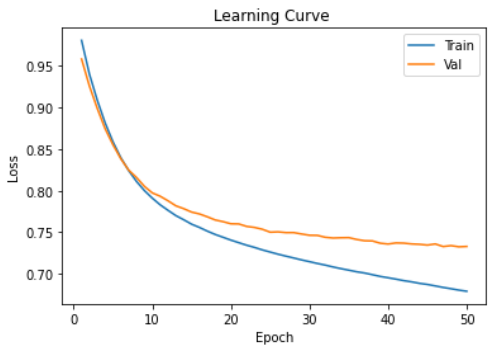
Βλέπουμε ότι αυξήθηκε ακόμα περισσότερο το overfitting από το προηγούμενο πείραμα χωρίς να έχει αλλάξει σημαντικά τα f1 scores, αυτό σημαίνει ότι το μοντέλο έχει γίνει πολύ περίπλοκο και δεν έχει την δυνατότητα να μάθει νέα δεδομένα.

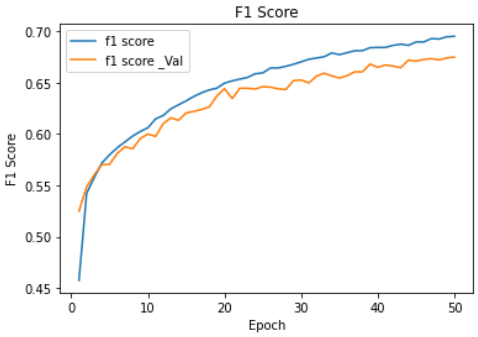
Multilayer Recurrent neural network

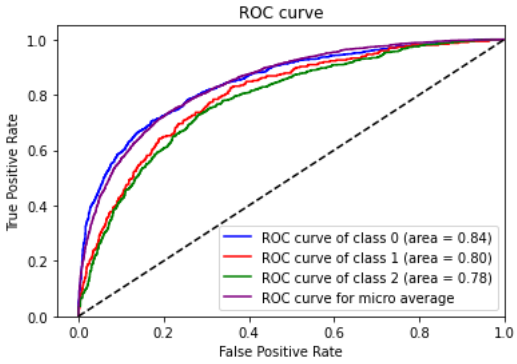
Τώρα θα προσπαθήσουμε να αυξήσουμε τον αριθμό των σωστών αποτελεσμάτων, χωρίς να υπάρχει overfitting με την χρήση multilayer RNN. Με αυτόν τον τρόπο το μοντέλο θα μαθαίνει περισσότερα και περίπλοκα δεδομένα αφού για κάθε επίπεδο θα δημιουργεί μια πιο σύνθετη αναπαράσταση των χαρακτηριστικών της τρέχουσας εισόδου. Οπότε αρχικά θα δοκιμάσουμε 2 επίπεδα με 128 hidden size για να το συγκρίνουμε με το παράδειγμα των 256 από πριν.

Training loss = 0.6790148 Validation loss = 0.7328859

Training F1_score = 0.6953515 Validation F1_score = 0.6751424





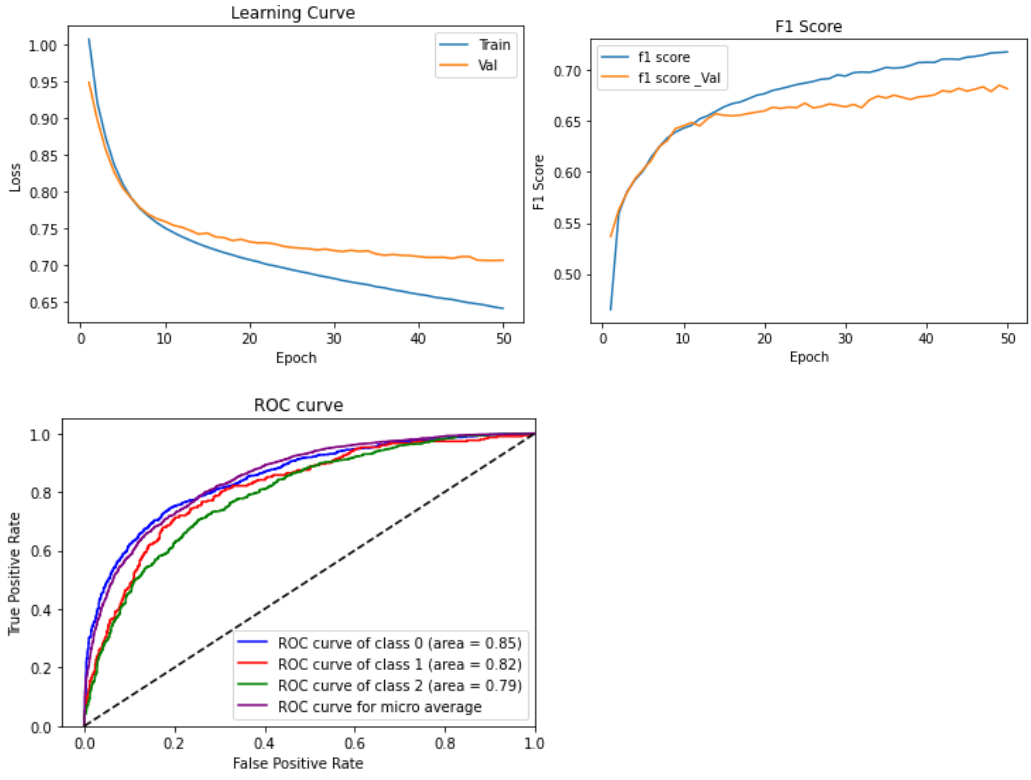


Παρατηρούμε ότι ενώ οι τιμές των f1 scores παρέμειναν οι ίδιες, το validation loss αυξήθηκε. Αυτό σημαίνει ότι το μοντέλο εμφανίζει overfitting για τον λόγω ότι η αύξηση της πολυπλοκότητας δεν επηρέασε πολύ τα δεδομένα.

Το ίδιο βλέπουμε και για 2 επίπεδα με 256 hidden size αν το συγκρίνουμε με το παράδειγμα των 512.

Training loss = 0.6417678 Validation loss = 0.7070683

Training F1_score = 0.7179624 Validation F1_score = 0.6818717

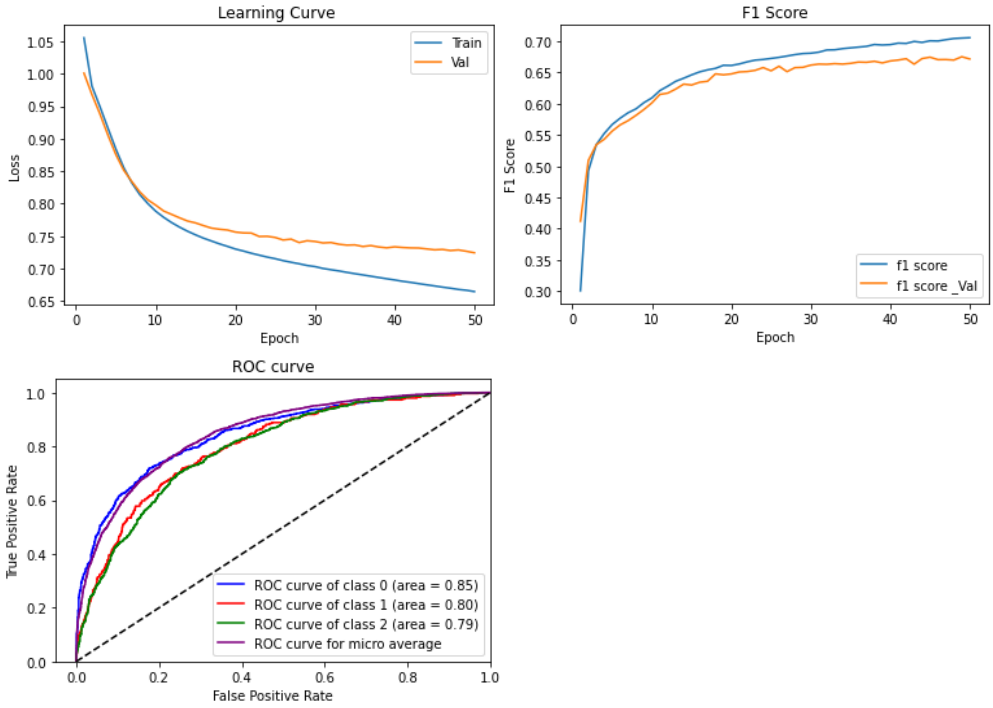


Δηλαδή οι τιμές των f1 score παραμένουν οι ίδιες ενώ αυτήν την φορά μειώνεται λίγο το training loss και εμφανίζει πάλι μεγαλύτερο overfitting.

Δοκιμάζουμε να αυξήσουμε τον αριθμό των επιπέδων κατά 1 για να παρατηρήσουμε αυτό το φαινόμενο. Οπότε για 3 επίπεδα με 128 size έχουμε:

Training loss = 0.6647568 Validation loss = 0.7244770

Training F1_score = 0.7058106 Validation F1_score = 0.6717790

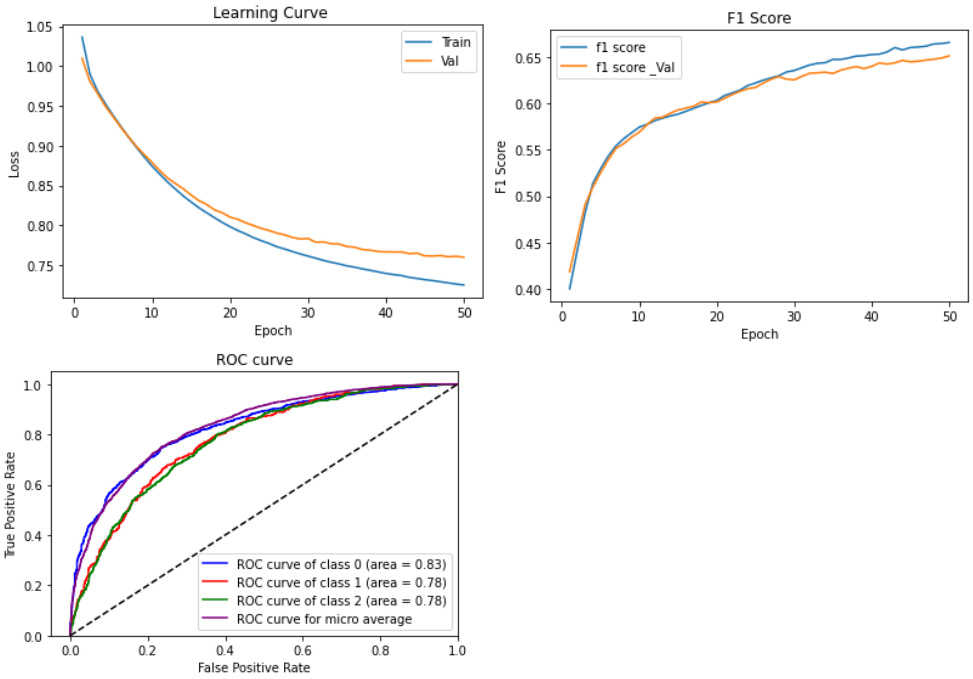


Βλέπουμε ότι μειώθηκαν τα f1 scores επειδή με την αύξηση των επιπέδων ταυτόχρονα αυξήθηκε και η πολυπλοκότητα οπότε χωρίς κάποια τεχνική regularization εμφανίζεται overfitting.

Για την μείωση του overfitting θα προσπαθήσουμε να μειώσουμε το hidden size και τα hidden layer παλι σε 2 ώστε το μοντέλο να είναι λίγο πιο απλό, οπότε για hidden size 64 βγάζουμε τις εξής τιμές:

Training loss = 0.7252391 Validation loss = 0.7601034

Training F1_score = 0.6660762 Validation F1_score = 0.6515494



Από τα αποτελέσματα αυτά παρατηρούμε ότι μειώνεται κάπως το overfitting αλλά ταυτόχρονα μειώθηκε σημαντικά και τα f1 scores. Με αυτό συμπεραίνουμε ότι πρέπει να βρούμε διαφορετικές τεχνικές για την εμφανίζεται overfitting όπως dropout.

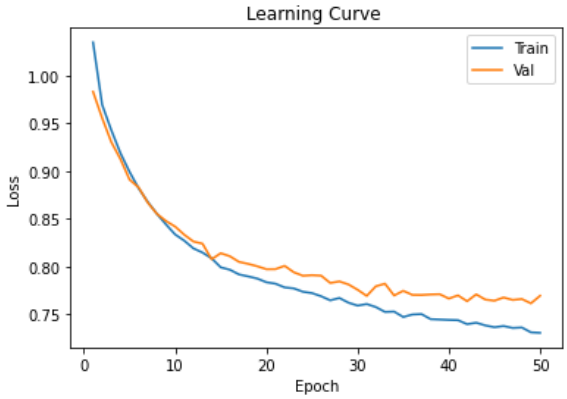
Dropout

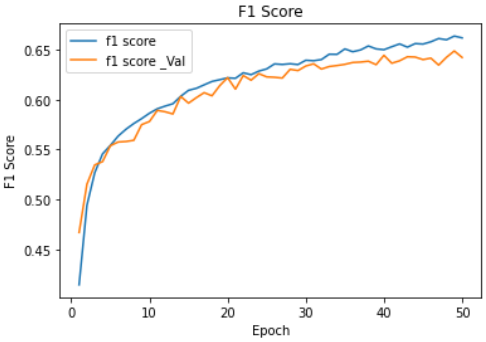
Με την χρήση dropout θα προσπαθήσουμε να κάνουμε regularization αφού οι είσοδοι και οι επαναλαμβανόμενοι σύνδεσμοι αποκλείονται πιθανοτικά από την ενεργοποίηση και τις ενημερώσεις βάρους κατά την εκπαίδευση του. Αυτό έχει ως αποτέλεσμα τη μείωση τις υπερπροσαρμογής και τη βελτίωση τις απόδοσης του.

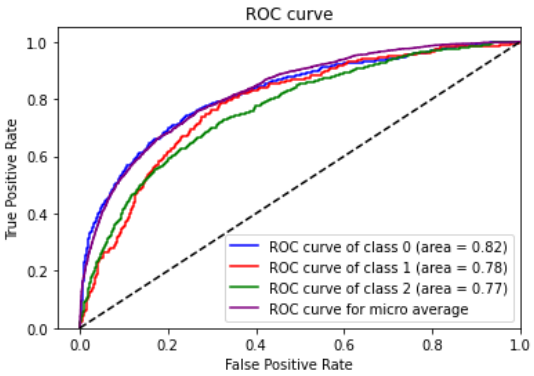
Οπότε για αρχή θα δοκιμάσουμε με 2 hidden layers μεγέθους 128 και dropout μεταξύ των επιπέδων να ισούται με 0.3.

Training loss = 0.7306360 Validation loss = 0.7693908

Training F1_score = 0.6619752 Validation F1_score = 0.6423692





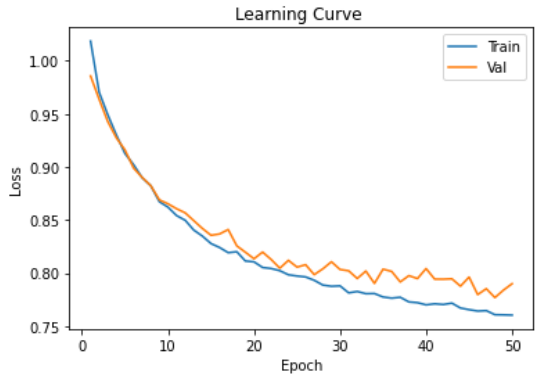


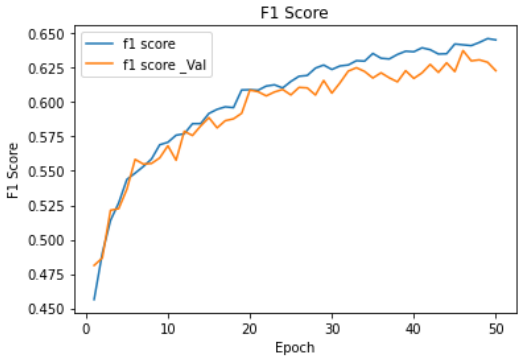
Συγκρίνοντας το παράδειγμα αυτό με εκείνο χωρίς dropout παρατηρούμε ότι η απόκλιση των losses άρα και το overfitting έχει κατέβει σημαντικά. Τις τα losses για train και validation έχουν αυξηθεί και ταυτόχρονα μειώθηκαν και τα f1 scores. Από το ROC curve παρατηρούμε ότι οι γραμμές των κλάσεων εμφανίζονται πολύ πιο κοντά μεταξύ τις από το προηγούμενο παράδειγμα που σημαίνει ότι αυτή την φορά δεν βρίσκει μια κλάση τις περισσότερες φορές από κάποια άλλη άρα και το μοντέλο τις είναι πιο συμμετρικό.

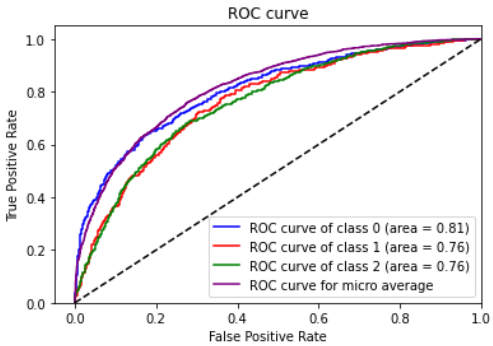
Πρέπει να πειραματιστούμε με τις τιμές του dropout για την εύρεση κάποιας που να μην μειώνει κατά πολύ τα scores. Αν αυξήσουμε το dropout σε 0.5 θα υπάρχει περισσότερο regularization οπότε πρέπει να δούμε ακόμα μικρότερη απόκλιση των losses και ακόμα χαμηλότερα f1 scores. Οπότε για 0.5 dropout έχουμε:

Training loss = 0.7603615 Validation loss = 0.7899742

Training F1_score = 0.6453426 Validation F1_score = 0.6229274





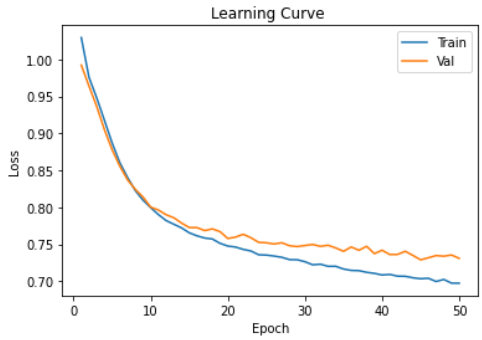


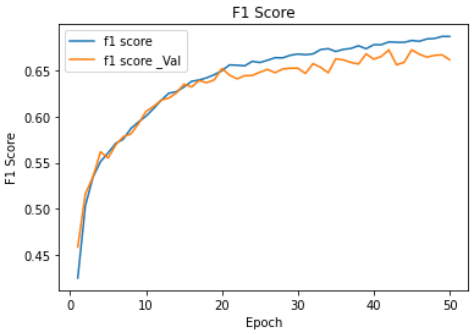
Από τα f1 scores και τις γράφους παρατηρούμε ότι το μοντέλο δυσκολεύεται να μάθει τα δεδομένα λόγω τις κανονικοποίησης. Οπότε η υπόθεση ήταν σωστή άρα καταλήγουμε ότι η χρήση του dropout πρέπει να γίνεται για χαμηλότερες τιμές.

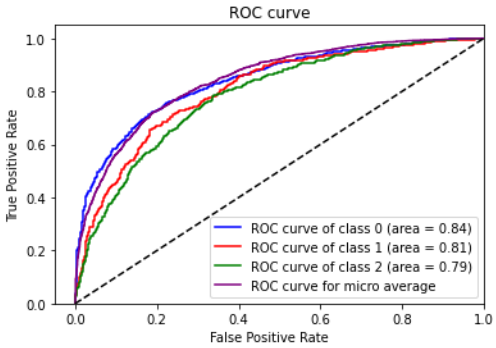
Οπότε θα πειραματιστούμε με ένα πιο περίπλοκο μοντέλο, τριών επιπέδων και μέγεθος 128, που έχουμε συμπεράνει ότι θα εμφανίσει overfitting αλλά αυτήν την φορά θα χρησιμοποιήσουμε dropout για κανονικοποίηση ώστε να βγάλει καλά f1 scores και χαμηλή απόκλιση των losses. Άρα για dropout 0.1 βρίσκουμε:

Training loss = 0.6974702 Validation loss = 0.7311237

Training F1_score = 0.6864884 Validation F1_score = 0.6613413







Τις υποθέσαμε το μοντέλο έχει αρκετά καλά f1 scores αλλά παρουσιάζει λίγο overfitting και από τις γράφους καταλαβαίνουμε ότι έχει μια μικρή δυσκολία μάθησης νέων δεδομένων. Παρόλα αυτά, είναι το πείραμα με τις καλύτερες αποδόσεις αφού εμφανίζεται αναλογικά το μεγαλύτερο f1 score με το μικρότερο loss μέχρι στιγμής.

RNN problems & gradient clipping

Τα RNN δίκτυα έχουν μερικά θεμελιώδη προβλήματα όπως το exploding gradient και το vanishing gradient.

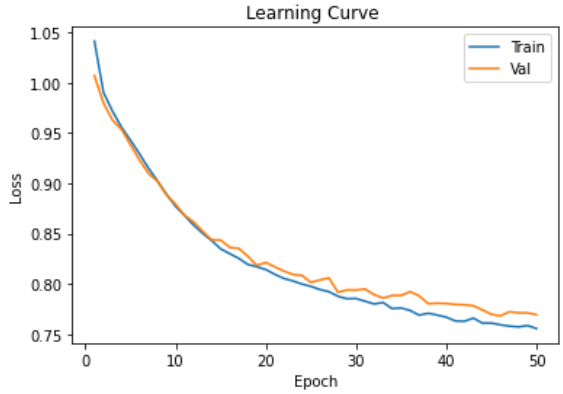
Το exploding gradient γίνεται όταν συσσωρεύονται μεγάλα gradients σφάλματος και έχουν ως αποτέλεσμα πολύ μεγάλες ενημερώσεις στα βάρη του μοντέλου κατά την εκπαίδευση. Αυτό έχει ως αποτέλεσμα το μοντέλο να μην μπορεί να μάθει νέα δεδομένα αφού τα προηγούμενα υπερτερούν πολύ αναλογικά στο βάρος. Μια λύση στο πρόβλημα αυτό είναι να χρησιμοποιήσουμε gradient clipping, κατά το οποίο αφού επιλέξουμε ένα threshold τότε όλες οι νόρμες που είναι υψηλότερες ή χαμηλότερες του threshold κατεβαίνουν στην νόρμα του βάλαμε.

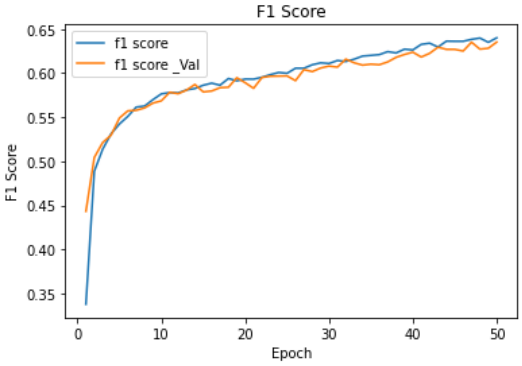
Ενώ το vanishing gradient συμβαίνει κατά το backpropagation όταν τα βάρη των προηγουμένων κόμβων χάνονται, δηλαδή οι τιμές φτάνουν κοντά στο μηδέν, οπότε το μοντέλο δεν μπορεί να θυμάται παλιά δεδομένα. Αυτό συμβαίνει επειδή η κλίση της συνάρτησης απώλειας μειώνεται εκθετικά με το χρόνο. Μια αρχική λύση είναι πάλι το gradient clipping για τον λόγω που προαναφέραμε.

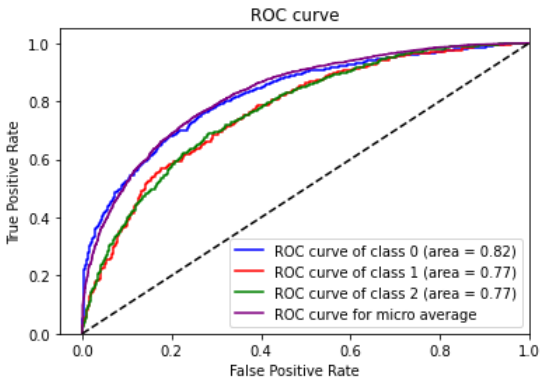
Οπότε θα δοκιμάσουμε το προηγούμενο πείραμα αλλά αυτήν την φορά θα χρησιμοποιήσουμε gradient clipping για να μειώσουμε τα προβλήματα αυτά. Άρα κάνουμε το threshold να ισούται με 10 e-8 και έχουμε τα εξής αποτελέσματα:

Training loss = 0.7557057 Validation loss = 0.7694345

Training F1_score = 0.6400489 Validation F1_score = 0.6352289







Παρατηρούμε ότι με gradient clipping εχεί μειωθεί κατά πολύ το overfitting αλλά αντι για αυτό μειώθηκαν τα f1 scores οπότε το μοντέλο έχει δυσκολία να μάθει νέα δεδομένα

LSTM

Για την επίλυση του vanishing gradient μπορούμε να χρησιμοποιήσουμε διαφορετικούς τύπους κελιών όπως LSTM και GRU.

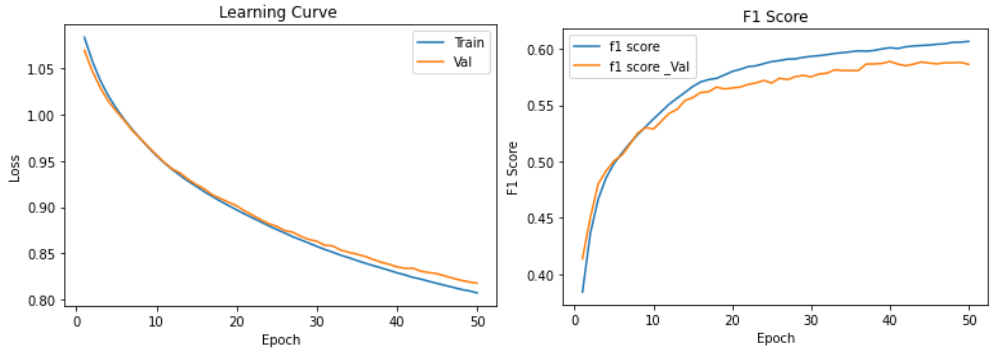
Τα LSTM είναι ένας τύπος RNN όπου οι κόμβοι περιλαμβάνουν ένα cell state, πέρα από το hidden state, που μπορεί να διατηρήσει πληροφορίες στη μνήμη για μεγάλες χρονικές περιόδους. Επίσης, ένα σύνολο πυλών χρησιμοποιείται για τον έλεγχο του πότε οι πληροφορίες εισέρχονται στη μνήμη, εξέρχονται και ξεχνιούνται.

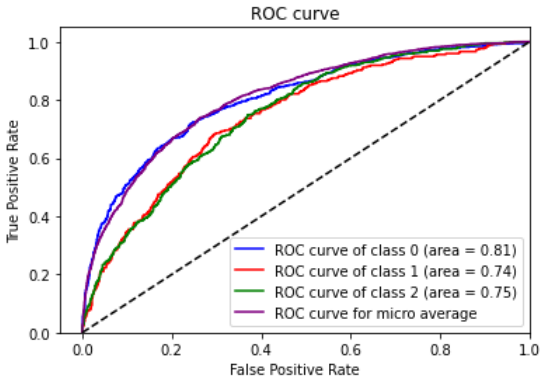
Οπότε θα ξεκινήσουμε τον πειραματισμό με LSTM δίκτυο χρησιμοποιώντας ένα απλό μοντέλο και θα το συγκρίνουμε με την συμπεριφορά του RNN.

Οπότε για LSTM με 1 επίπεδο και 32 hidden size έχουμε:

Training loss = 0.8075987 Validation loss = 0.8180161

Training F1_score = 0.6067188 Validation F1_score = 0.5863962



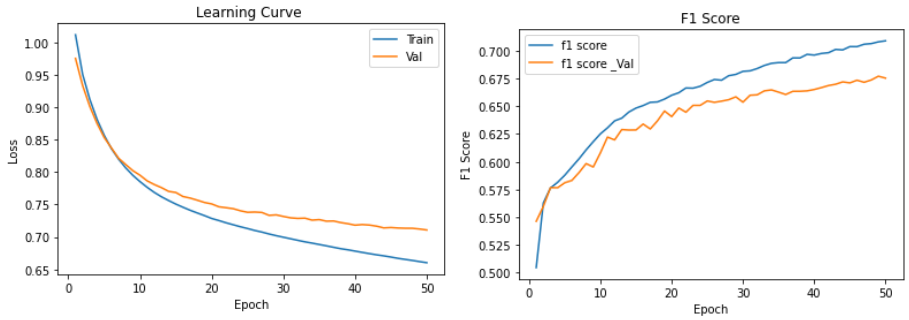


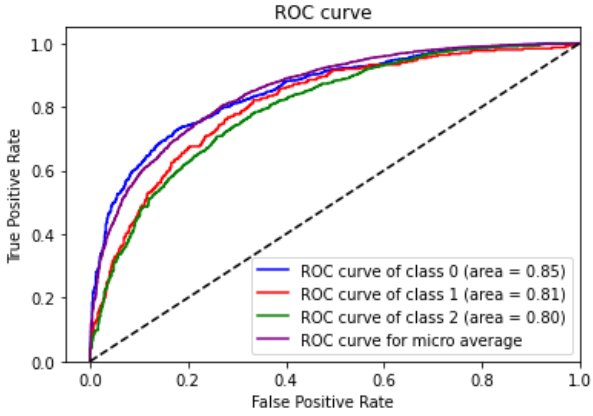
Συγκρίνοντας με το RNN βλέπουμε ότι τα losses για training και validation είναι λίγο χαμηλότερα και έχουν μικρότερη απόκλιση, οπότε και λιγότερο overfitting. Επίσης, τα f1 scores είναι σχετικά υψηλότερα στο LSTM οπότε το μοντέλο αυτό είναι και πιο ακριβής. Όμως τα νούμερα δεν αλλάζουν κατά πολύ για τον λόγω ότι το μοντέλο είναι τόσο απλό που δεν χρειαζόντουσαν κατά μεγάλο βαθμό τα memory cells. Οπότε θα προσπαθήσουμε για ένα μεγαλύτερο δίκτυο.

Άρα για LSTM με 1 επίπεδο και 256 μέγεθος έχουμε:

Training loss = 0.6603513 Validation loss = 0.7107653

Training F1_score = 0.7088871 Validation F1_score = 0.6753253



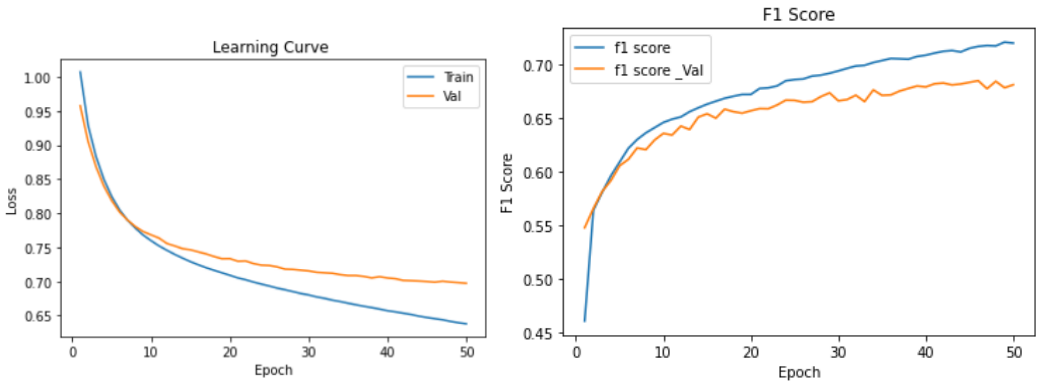


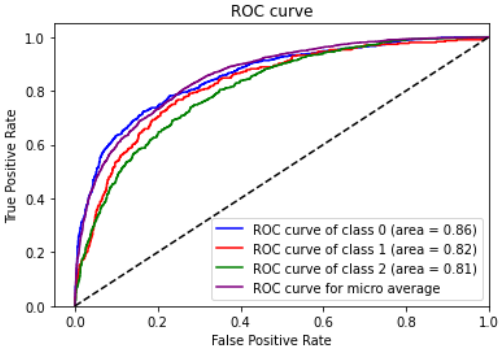
Πάλι παρατηρούμε την ίδια συμπεριφορά, ότι δηλαδή το LSTM παρουσιάζει μεγαλύτερη ακρίβεια και ταυτόχρονα έχει λιγότερο overfitting από τον RNN.

Επίσης θα δοκιμάσουμε για LSTM με 1 επίπεδο και μέγεθος 512 βρίσκουμε τα εξής:

Training loss = 0.6380127 Validation loss = 0.6973812

Training F1_score = 0.7199916 Validation F1_score = 0.6810882





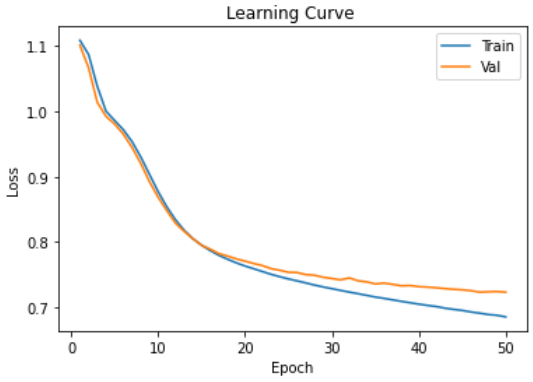
Παρατηρούμε όπως και στο προηγούμενο παράδειγμα ότι ταυτόχρονα με την αύξηση των f1 scores ταυτόχρονα χαμηλώθηκαν και τα losses. Αυτό σημαίνει ότι πάλι με την χρήση LSTM το μοντέλο δεν εμφανίζει overfitting και είναι πιο ακριβής.

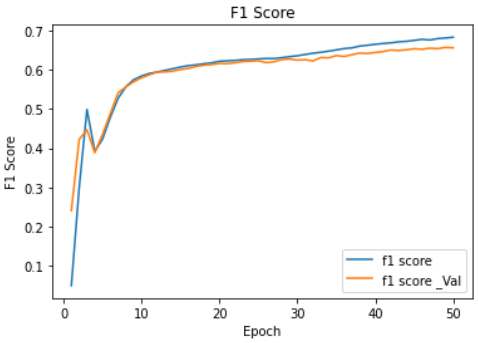
Multilayer LSTM

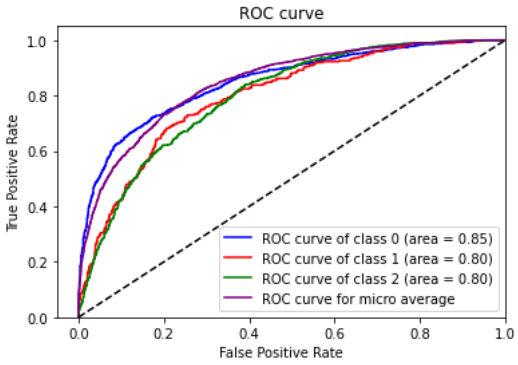
Αφου καταλήξαμε ότι ο LSTM δουλεύει καλύτερα από τον RNN και σε μεγαλύτερα δίκτυα, τώρα θα συγκρίνουμε τον LSTM με μεγάλη πολυπλοκότητα δικιμάζοντας τον για 3 επίπεδα με 128 μέγεθος.

Training loss = 0.6853122 Validation loss = 0.7231908

Training F1_score = 0.6826551 Validation F1_score = 0.6557700





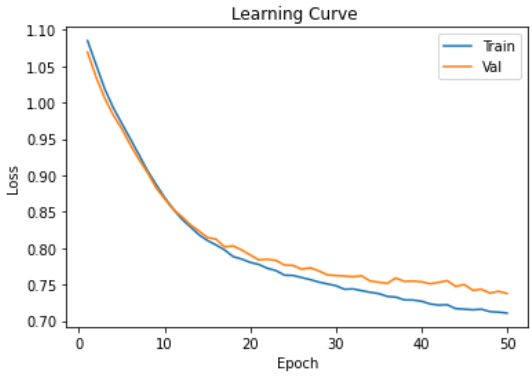


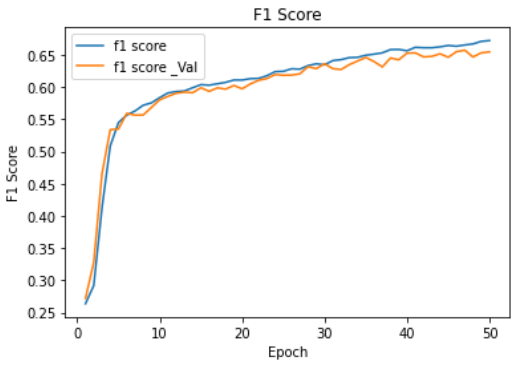
Βλέπουμε χρησιμοποιώντας LSTM το μοντέλο δεν παρουσιάζει κάποια μεγάλη διαφορά στο στα losses και έχει χαμηλότερα f1 scores οπότε δεν μπορεί να μάθει τόσο πολλά δεδομένα. Οπότε μπορεί να χρειάζεται κάποια τεχνική μείωσης του overfitting όπως dropout και ύστερα θα δοκιμάσουμε gradient clipping για την αποφυγή του exploding gradient.

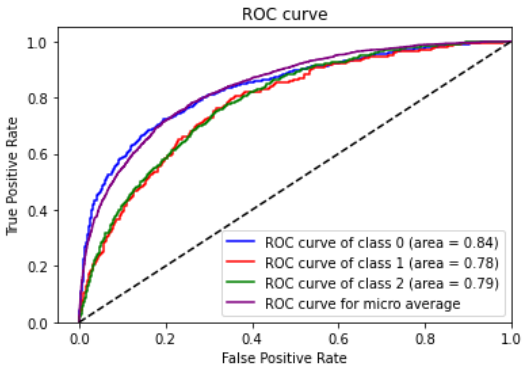
Οπότε για 2 επίπεδα με μέγεθος 128 και dropout 0.3 έχουμε:

Training loss = 0.7105973 Validation loss = 0.7376332

Training F1_score = 0.6722384 Validation F1_score = 0.6546035





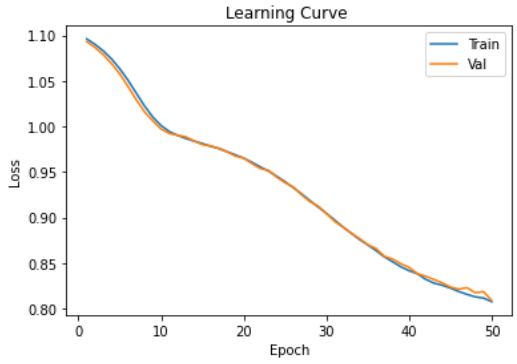


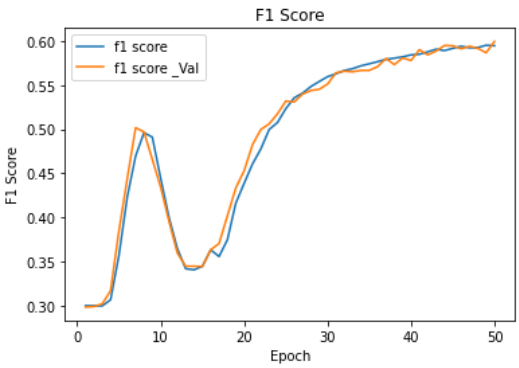
Συγκρίνοντας με το RNN παρατηρούμε ότι το overfitting έχει χαμηλώσει πάρα πολύ αφού το validation loss κατέβηκε αρκετά. Αυτό σημαίνει ότι στο συγκεκριμένο παράδειγμα εμφανιζόταν το πρόβλημα με τα vanishing gradients το οποίο LSTM έφτιαξε σε μεγάλο βαθμό. Ταυτόχρονα ανέβηκαν τα f1 scores για τον ίδιο λόγω. Οπότε το μοντέλο συμπεριφέρεται καλά όταν χρησιμοποιούμε regularization.

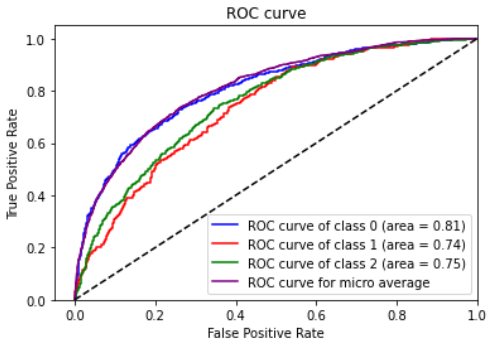
Τώρα θα δοκιμάσουμε το τελευταίο παράδειγμα του RNN που ήταν 3 επίπεδα με μέγεθος 128, dropout 0.1 και gradient clipping 10 e-8.

Training loss = 0.8074359 Validation loss = 0.8086857

Training F1_score = 0.5945431 Validation F1_score = 0.5994450





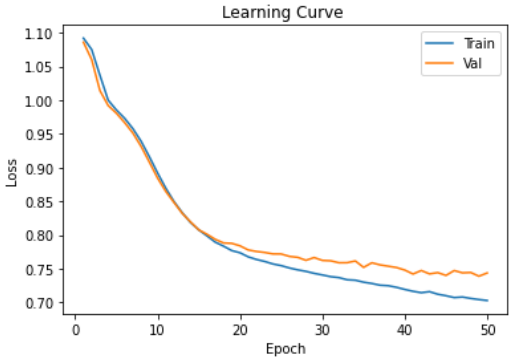


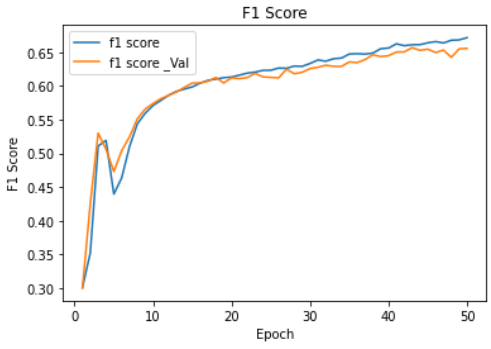
Παρατηρούμε ότι τα f1 scores έχουν πέσει πολύ ενώ τα losses έχουν ανέβει αρκετά. Αυτό σημαίνει ότι εμφανίζει underfitting αφού αδυνατεί να μάθει τα δεδομένα μας. Υποθέτουμε ότι η αιτία ήταν η μικρή τιμή στο threshold που βάλαμε για gradient clipping και αυτό είχε σαν αποτέλεσμα το μοντέλο να μην μπορεί να δημιουργήσει σωστά βάρη για τους κόμβους.

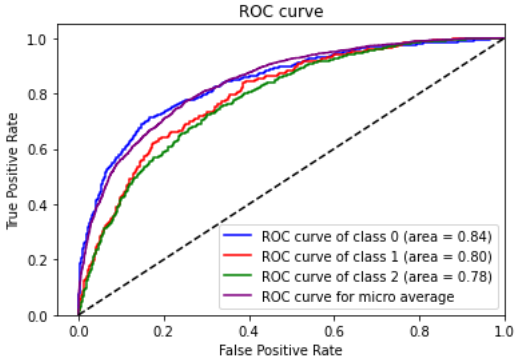
Οπότε θα δοκιμάσουμε χωρίς gradient clipping και επιστρέφει τα εξής αποτελέσματα.

Training loss = 0.7031019 Validation loss = 0.7438978

Training F1_score = 0.6718797 Validation F1_score = 0.6558448







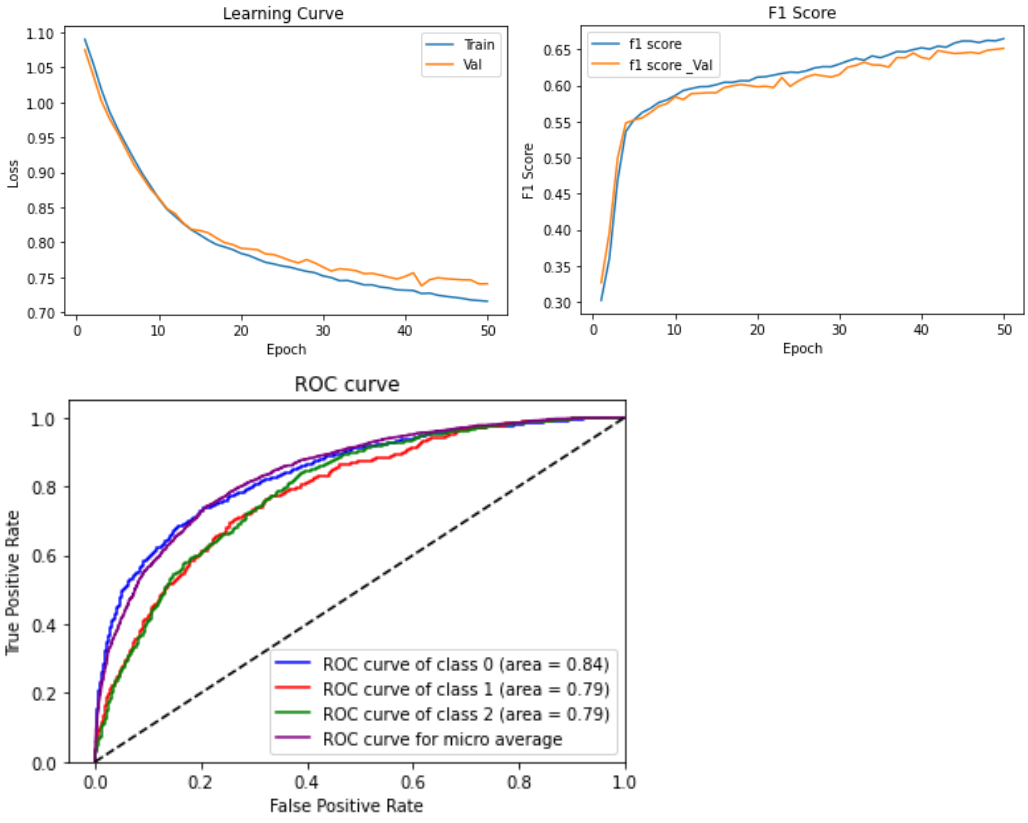
Με την αφαίρεση του gradient clipping βλέπουμε ότι έφτιαξε το πρόβλημα με το underfitting αλλά εμφάνισε παρόμοια αποτελέσματα με τα 2 επίπεδα και dropout 0.3 αλλά με περισσότερο overfitting, οπότε θα επιλέξουμε διαφορετικό μοντέλο με το RNN.

BIDIRECTIONAL

Θα δοκιμάσουμε να κάνουμε LSTM bidirectional, ώστε τα inputs να επεξεργάζονται όχι μόνο προς τα εμπρός αλλά και αντίθετα. Η τεχνική αυτή βοηθάει πολύ τα δεδομένα που βρίσκονται με μια συγκεκριμένη σειρά ώστε να τα αναλύει και από τις 2 μεριές. Οπότε χρησιμοποιώντας το παράδειγμα με 2 επίπεδα με μέγεθος 128 και dropout 0.3 έχουμε:

Training loss = 0.7159636 Validation loss = 0.7409537

Training F1_score = 0.6646791 Validation F1_score = 0.6512452



Από το προηγούμενο παράδειγμα με LSTM δεν παρατηρούμε κάποια μεγάλη διαφορά στα losses αλλά το f1 score για το training χαμηλώθηκε κατά 0.01 μονάδα. Αυτό σημαίνει ότι το bidirectional κάνει πιο περίπλοκο το μοντέλο χωρίς να κερδίζει τόσο σε ακρίβεια.

GRU cells

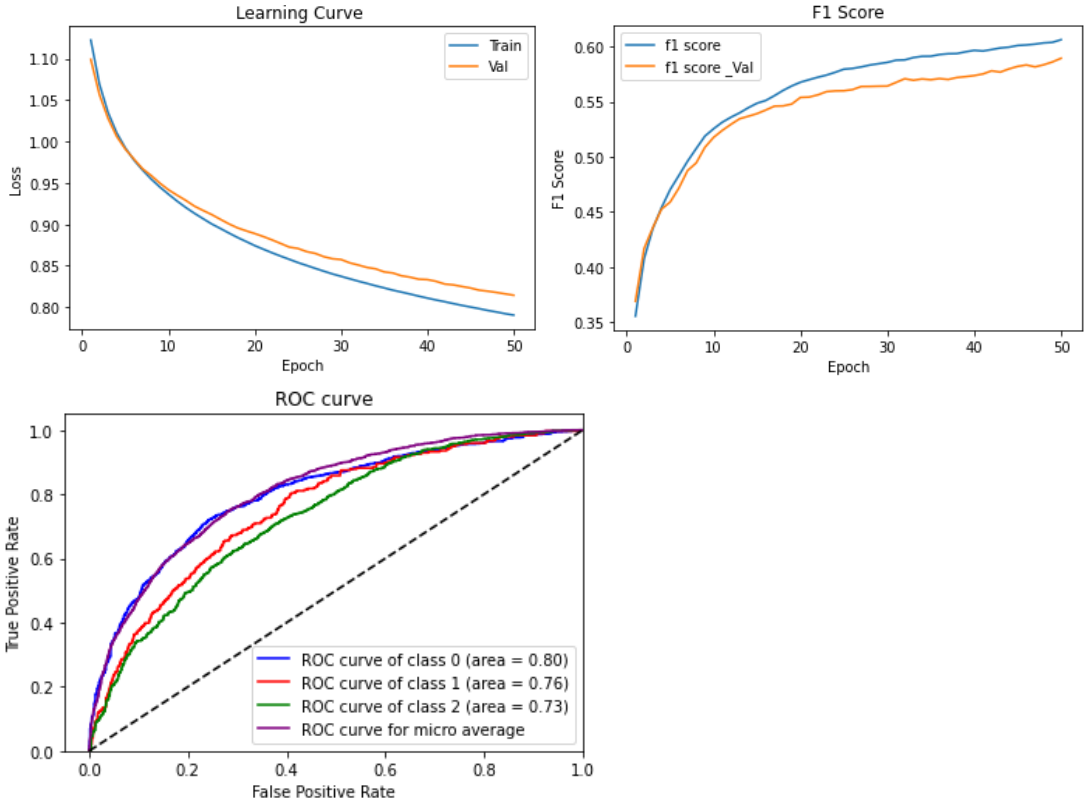
Η αιτία που επιστρέφει σχετικά χαμηλά f1 scores ενώ ταυτόχρονα εμφανίζει λίγο overfitting μπορεί να βρίσκεται στο περίπλοκο μοντέλο του LSTM. Οπότε αν έχουμε λίγα σχετικά δεδομένα πρέπει να προτιμήσουμε ένα μοντέλο πιο απλοϊκό, όπως το GRU.

Το GRU δίκτυο θυμίζει πολύ το LSTM μοντέλο με την διαφορά ότι πλέον δεν έχει ξεχωριστό memory cell αλλά χρησιμοποιεί το hidden state για να μεταφέρει δεδομένα. Επίσης, έχει μόνο 2 gets, για reset και update.

Οπότε θα ξεκινήσουμε τον πειραματισμό με GRU μοντέλο, συγκρίνοντας ένα πολύ απλό σύστημα με 1 επίπεδο και 32 hidden size με το LSTM. Οπότε έχουμε τα εξής αποτελέσματα:

Training loss = 0.7904651 Validation loss = 0.8144698

Training F1_score = 0.6062167 Validation F1_score = 0.5893660

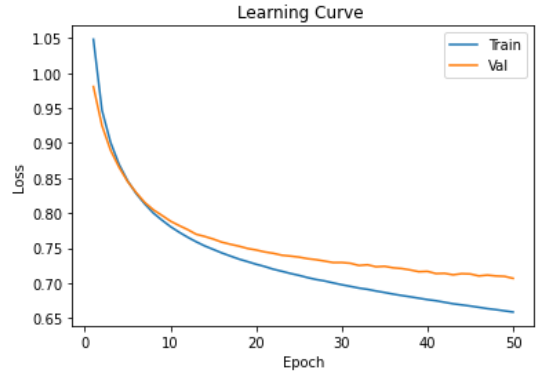


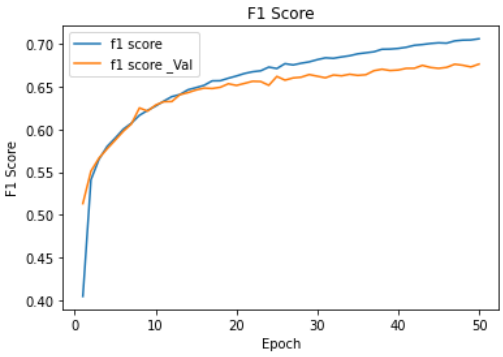
Παρατηρούμε ότι τα f1 scores όπως και το losses δεν διαφέρουν πολύ από εκείνα του LSTM αφού το μοντέλο είναι πολύ απλό για χρησιμοποιήσει τα memory cells του LSTM ή τα ειδικά hidden states του GRU, οπότε χρειαζόμαστε ένα πιο σύνθετο μοντέλο για την παρατήρηση του.

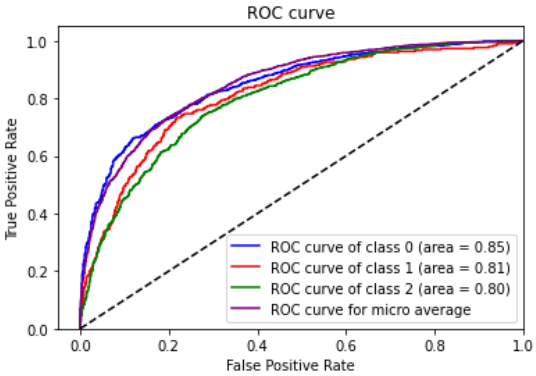
Οπότε για 1 επίπεδο και μέγεθος 256 αντίστοιχα έχουμε:

Training loss = 0.6584566 Validation loss = 0.7066432

Training F1_score = 0.7064450 Validation F1_score = 0.6767661







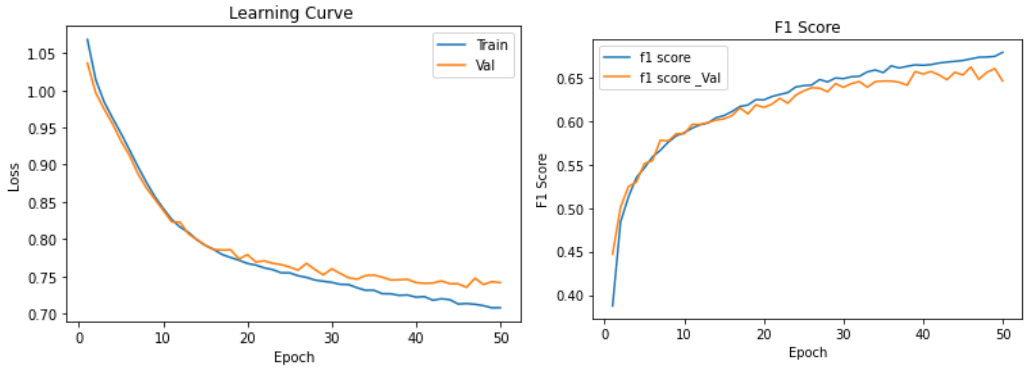
Συγκρίνοντας με τον LSTM βλέπουμε πως έχει μειωθεί λίγο το overfitting και τα losses των train και validation set ενώ τα f1 scores έχουν παραμείνει τα ίδια. Με αυτό συμπεραίνουμε ότι το GRU μπορεί να παρουσιάζει λιγότερο overfitting για μεγαλύτερα μοντέλα από το LSTM. Αλλά για πιο περίπλοκες δομές πρέπει να εμφανίζεται πρόβλημα στα losses αφού έχει μεγάλο dataset, οπότε και πολλές πληροφορίες.

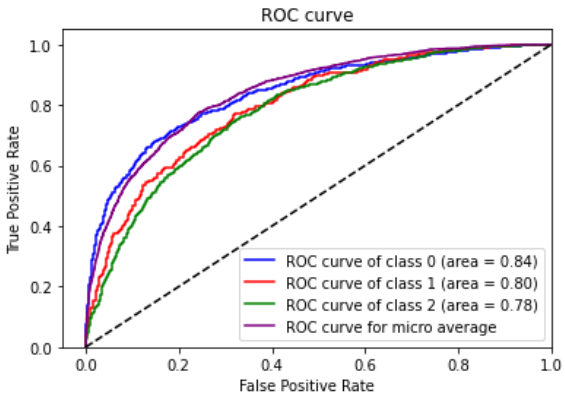
Multilayer GRU

Για το σιγουρέψουμε όμως θα χρησιμοποιήσουμε multilayer GRU των 2 επιπέδων με μέγεθος 128 και dropout 0.3, οπότε έχουμε:

Training loss = 0.7077786 Validation loss = 0.7414679

Training F1_score = 0.6786329 Validation F1_score = 0.6457351



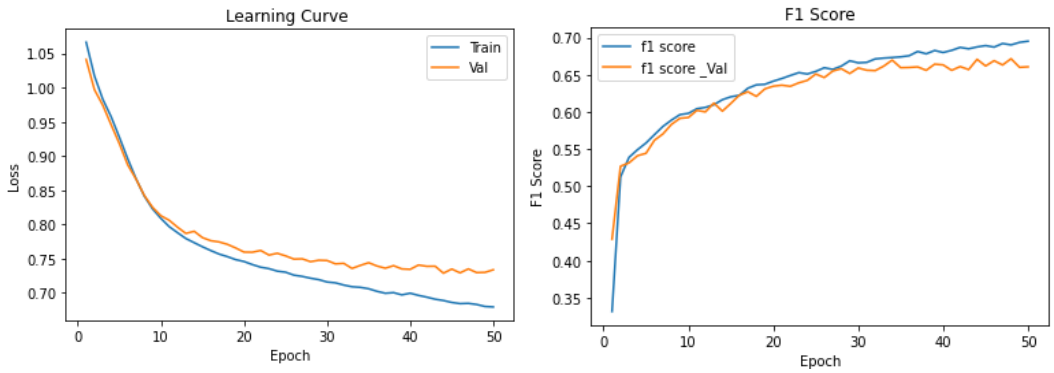


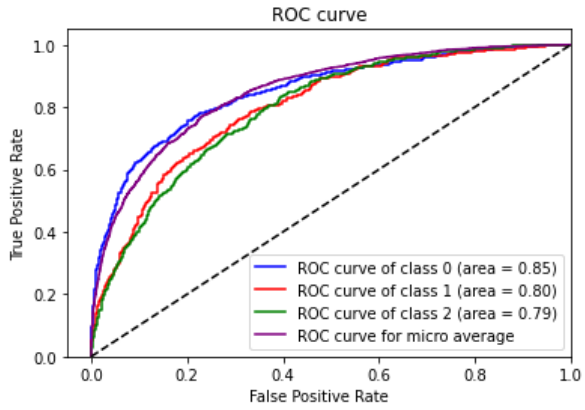
Όπως υποθέσαμε το μοντέλο με την χρήση GRU εμφανίζει χαμηλότερη ακρίβεια από εκείνο με LSTM αφού το μοντέλο αυτό δεν έχει memory cells οπότε και είναι πιο απλό.

Τέλος θα δοκιμάσουμε να τρέξουμε για 3 επίπεδα και μέγεθος 128 με dropout 0.1 ώστε να το συγκρίνουμε με τον LSTM

Training loss = 0.6795236 Validation loss = 0.7338082

Training F1_score = 0.6949969 Validation F1_score = 0.6605430





Παρατηρούμε ότι συγκριτικά με το LSTM συμπεριφέρεται παρόμοια ένα RNN μοντέλο, δηλαδή εμφανίζει μεγαλύτερο overfitting, γιατι οι γραμμές είναι απομακρυσμένες στον learning curve graph αλλά έχει μια αρκετή αύξηση για στα f1 scores και ειδικότερα του training set. Αυτό σημαίνει ότι χρησιμοποιώντας πιο απλό cell type, δηλαδή GRU, δημιουργεί αρκετό overfitting, ενώ ταυτόχρονα είναι λίγο πιο ακριβής στα αποτελέσματά του.

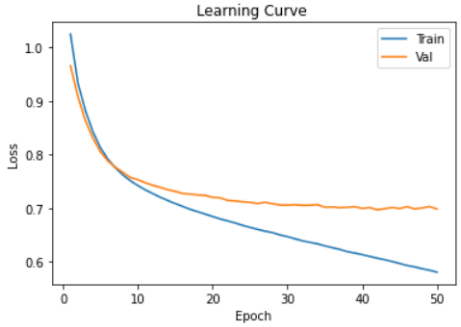
Skipping Connections

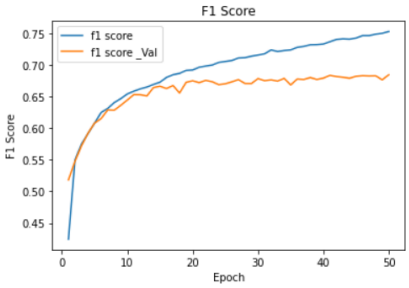
Οπότε καταλήξαμε ότι για τα δικά μας δεδομένα τα καλύτερα αποτελέσματα τα έχει το LSTM, οπότε στην συνέχεια θα ασχοληθούμε μόνο με αυτό. Τώρα θα δοκιμάσουμε να εισάγουμε skip connections στο μοντέλο μας και συγκεκριμένα ResNet skip connections.

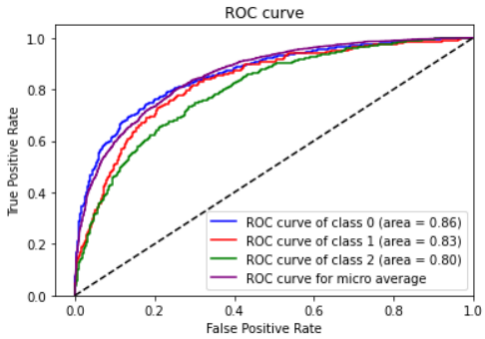
Αρχικά θα δούμε με μόνο 2 επίπεδα και μέγεθος 256 την συμπεριφορά των skip connections. Οπότε έχουμε τα εξής:

Training loss = 0.5804012 Validation loss = 0.6983669

Training F1_score = 0.7526480 Validation F1_score = 0.6843147





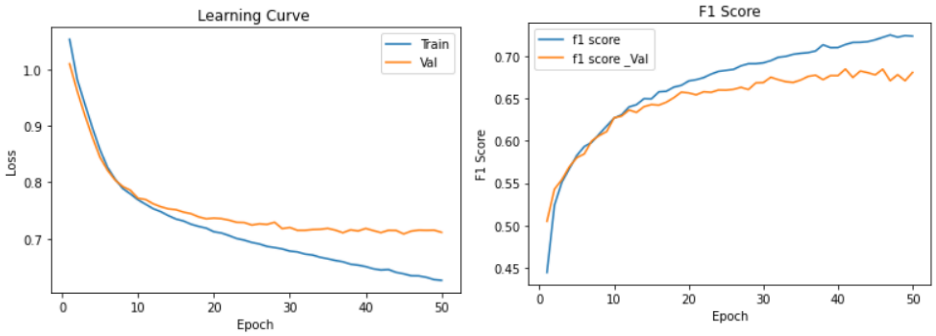


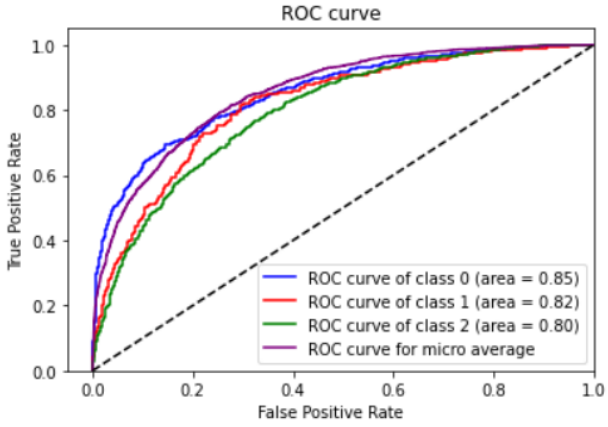
Παρατηρούμε ότι με την εισαγωγή των συγκεκριμένων skip connections το μοντέλο ενώ μαθαίνει γρήγορα τα δεδομένα, παρουσιάζει πολύ μεγάλο βαθμό overfitting και αδυνατεί να μάθει περισσότερες τιμές. Η αιτία είναι ότι με την χρήση λίγων επιπέδων το μοντέλο δεν παρουσιάζει την ανάγκη να παραλείψει κάποιο από αυτά, οπότε η τεχνική αυτή βοηθάει και πιο περίπλοκα μοντέλα.

Για αυτόν τον λόγω θα δοκιμάσουμε με 4 επίπεδα και μέγεθος 128 :

Training loss = 0.6263741 Validation loss = 0.7113764

Training F1_score = 0.7234862 Validation F1_score = 0.6806306





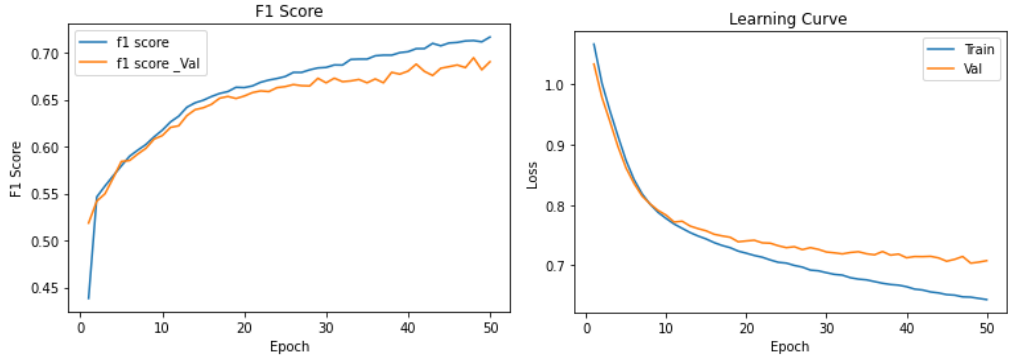
Βλέπουμε πάλι αρκετό overfitting αλλά σε μικρότερο βαθμό με τα λιγότερα επίπεδα, οπότε καταλήγουμε ότι δεν χρειάζεται να χρησιμοποιήσουμε skip connections για τα δεδομένα αυτά.

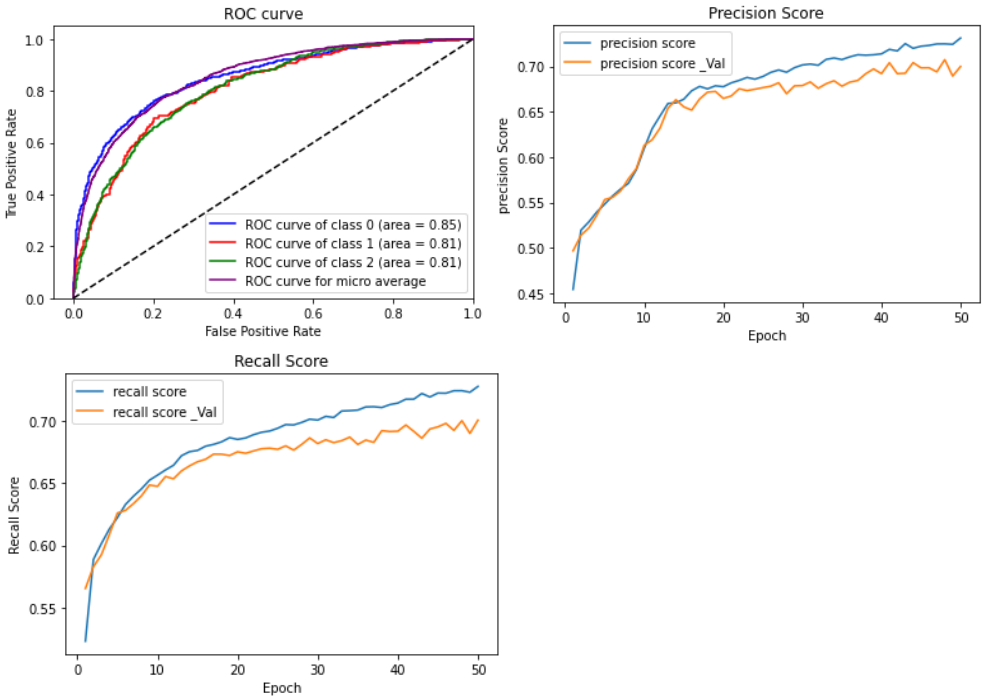
Καλύτερο Μοντέλο

Μετα από πειραματισμούς καταλήξαμε ότι τα καλύτερα αποτελέσματα τα επιστρέφει ένα μοντέλο LSTM με 2 επίπεδα με μέγεθος 256 με dropout 0.1. Δοκιμάζοντας όλες τις άλλες τεχνικές, οι τιμές των f1 score για train και validation χαμήλωναν, οπότε καταλήξαμε σε αυτό το πολύ απλό μοντέλο. Συγκεκριμένα παρέχει τα εξής αποτελέσματα:

Training loss = 0.6430181 Validation loss = 0.7076186

Training F1_score = 0.7166359 Validation F1_score = 0.6903352





Από τα γραφήματα και τις μετρήσεις είναι καθαρό ότι το μοντέλο αυτό είναι και το καλύτερο μέσα από τα πειράματα, αφού για το validation και training το f1 score φτάνει το 0.7. Όμως, και πάλι παρουσιάζει μια τάση για overfitting που δεν μπορούμε να φτιάξουμε και ταυτόχρονα να κρατήσουμε την ακρίβεια του για το validation set.

Σύγκριση

Το μοντέλο αυτό επιστρέφει παρόμοια αποτελέσματα με αυτά των προηγούμενων εργασιών, το οποίο είναι λογικό αφού ενώ κάναμε το μοντέλο μας πιο σύνθετο για να καταλαβαίνει τα δεδομένα μας, δεν πειραματιστήκαμε με τον optimizer και loss function για να βρούμε το βέλτιστο μοντέλο. Παρόλα αυτά, τα RNN δίκτυα και συγκεκριμένα τα LSTM δείχνουν μεγαλύτερη προοπτική από τα απλά feed forward μοντέλα. Τέλος, το ίδιο ακριβώς συμβαίνει και το softmax regression της πρώτης εργασίας.

Δεύτερο υποερώτυμα

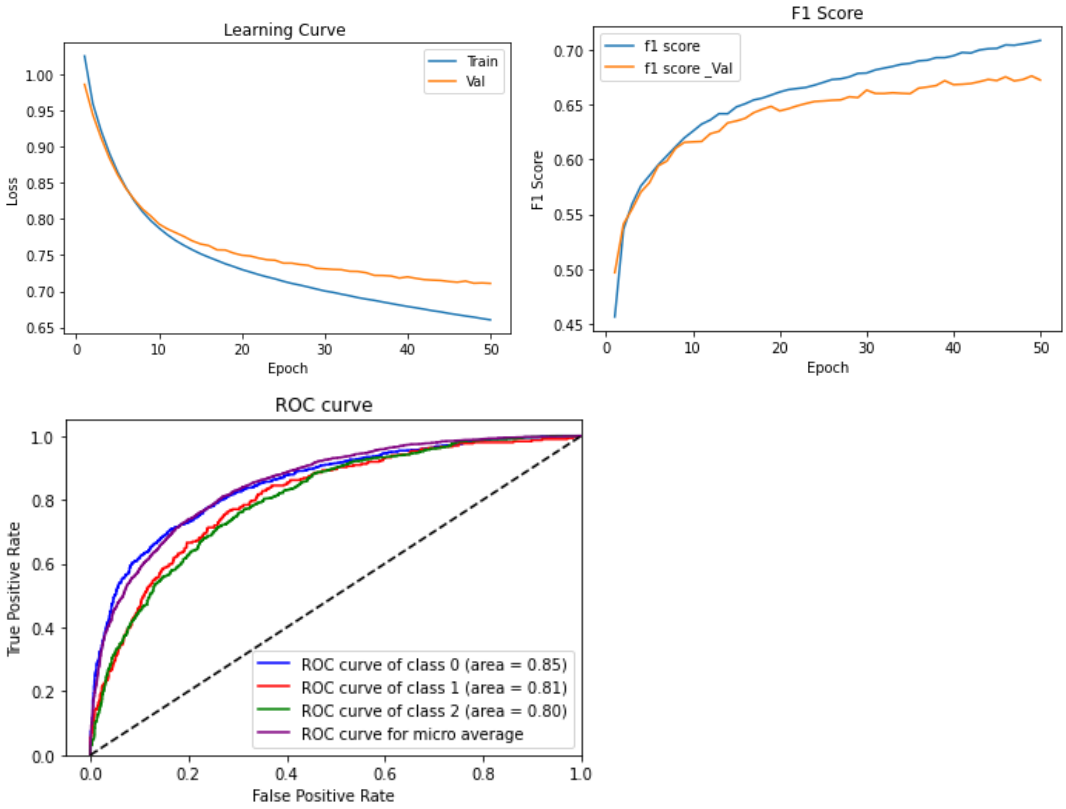
Για το δεύτερο υποερώτυμα πρέπει να προσθέσουμε attention στο μοντέλο από την πρώτο κομμάτι της εργασίας.

Το attention είναι ένας μηχανισμός στα RNN δίκτυα που το βοηθάνε να δίνουν προσοχή σε συγκεκριμένο κομμάτι των εισαγώμενων τιμών όταν προβλέπει ένα σημείο των εξαγώμενων τιμών, αυτό έχει σαν αποτέλεσμα το νευρωνικό δίκτυο να έχει την δυνατότητα να θυμάται πολλά δεδομένα.

Για την εργασία θα χρησιμοποιήσουμε attention σε ένα απλό μοντέλο LSTM με 1 επίπεδο μεγέθους 256, το οποίο επιστρέφει πολύ καλές τιμές τιμές f1 scores. Δεν χρησιμοποιούμε ένα multilayer μοντέλο για τον λόγω ότι παρουσίαζε αρκετό overfitting. Οπότε με την χρήση attention έχουμε τα εξής αποτελέσματα:

Training loss = 0.6605440 Validation loss = 0.7109654

Training F1_score = 0.7087398 Validation F1_score = 0.6725994



Ενώ το μοντέλο χωρίς attention επέστρεφε τα εξής δεδομένα.

Training loss = 0.6876106 Validation loss = 0.7315809

Training F1_score = 0.6886901 Validation F1_score = 0.6635800

Παρατηρούμε ότι με την προσθήκη attention έχουν κατέβει τα losses στο train και όπως στο validation set ενώ ταυτόχρονα ανέβηκαν αρκετά τα f1 scores. Αυτό γίνεται αφού με την χρήση attention το δίκτυο μπορεί να συγκεντρώνεται σε λιγότερα δεδομένα οπότε να παρουσίαζει και μεγαλύτερη ακρίβεια.