Νευρωνικά Δίκτυα - Βαθιά Μάθηση

Τμήμα Πληροφορικής ΑΠΘ – 7ο Εξάμηνο

Χειμώνας 2021-22

Υλοποίηση Radial Basis Function (RBF) NN για επίλυση προβλήματος παλινδρόμησης (regression)

Σπυράκης Άγγελος (9352)

aspyrakis@ece.auth.gr

Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Η/Υ

Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης

1 Εισαγωγή

Στην πρώτη εργασία του μαθήματος παρουσιάστηκαν τα αποτελέσματα εφαρμογής κατηγοριοποιητών Nearest Centroid, k-Nearest Neighbor, Multi-Layer Perceptron και Convolutional Neural Network στη βάση δεδομένων CIFAR-10, όπου η σύγκριση έδειξε ότι το καλύτερο μοντέλο ήταν αυτό ενός CNN. Έπειτα στην 2η εργασία υλοποιήθηκε ένας SVM κατηγοριοποιητής με διαφορετικές συναρτήσεις πυρήνα, όπου η καλύτερη συνάρτηση για το Landsat Satelite dataset ήταν η RBF.

Σε αυτή την εργασία, θα υλοποιήσουμε ένα **Radial Basis Function** νευρωνικό δίκτυο, το οποίο αποτελεί ένα συγκεκριμένο τύπο NN με **3 συνολικά στρώματα**. Αυτή τη φορά, όμως, θα επιλυθεί **πρόβλημα παλινδρόμησης** (regression) και όχι κατηγοριοποίησης. Το dataset που επιλέχθηκε είναι το **Airfoil Self-Noise**[1] της NASA, το οποίο αποτελείται από 1503 δείγματα και 6 features:

1. Frequency (Hz)
2. Angle of attack (degrees)
3. Chord length (m)
4. Free-stream velocity (m/s)
5. Suction side displacement thickness (m)
6. **Output:** Scaled sound pressure level (dB)

Όλα τα features είναι συνεχείς μεταβλητές και όχι κατηγορικές, και το ίδιο ισχύει για την έξοδο. Επομένως, ενώ συνήθως γινόταν μια αρχική επίλυση του προβλήματος εφαρμόζοντας τις τεχνικές nearest-centroid και k-NN, εδώ θα χρησιμοποιηθεί μόνο ο **k-NN regressor**, αφού η nearest-centroid αφορά μόνο προβλήματα κατηγοριοποίησης.

2 k-Nearest Neighbor

Αρχικά θα ελεγχθεί η απόδοση του k-NN regressor, ώστε να συγκριθεί αργότερα με την απόδοση του RBF νευρωνικού δικτύου. Ο κώδικας είναι ίδιος με αυτόν της δεύτερης εργασίας, με τη διαφορά ότι τώρα χρησιμοποιείται ο **KNeighborsRegressor** και όχι ο Classifier, της βιβλιοθήκης sklearn[2].

Αξίζει να σημειωθεί, ότι επειδή τα features είναι συνεχείς μεταβλητές, διαφορετικές μεταξύ τους, γίνεται **κανονικοποίηση** κάθε feature ξεχωριστά, αφαιρώντας την μέση τιμή και διαιρώντας με την τυπική απόκλιση (του κάθε feature). Επίσης, επειδή πλέον έχουμε πρόβλημα παλινδρόμησης, δεν μπορεί να χρησιμοποιηθεί η μετρική accuracy, αλλά αντ' αυτού χρησιμοποιούνται οι μετρικές Root Mean Squared Error (**RMSE**) και Coefficient of Determination (**R^2**). Και οι δύο μετρικές χρησιμοποιούνται συχνά για την αξιολόγηση ενός μοντέλου, με τη διαφορά ότι η τιμές της RMSE εξαρτώνται από το dataset, ενώ η R^2 έχει πάντα μέγιστη τιμή

[1] https://archive.ics.uci.edu/ml/datasets/airfoil+self-noise
[2] https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html

ίση με 1, το οποίο σημαίνει φυσικά τέλεια παλινδρόμηση. Τιμές μικρότερες του 0.4-0.5 συνήθως δείχνουν κακή απόδοση, ενώ ο συντελεστής μπορεί να λάβει και αρνητικές τιμές. Οι δύο μετρικές συνδέονται, αλλά αυτό θα παρουσιαστεί στην επόμενη ενότητα όπου θα γίνει λόγος για το MSE.

Για την εύρεση του βέλτιστου k γίνονται πειράματα με διαφορετικές τιμές, και τα αποτελέσματα παρουσιάζονται στον παρακάτω πίνακα.

k	RMSE	R^2
1	2.9439	0.7980
2	**2.4677**	**0.8580**
3	2.8148	0.8153
4	3.0930	0.7770
10	3.6624	0.6873

Πίνακας 1 - Ακρίβεια για διαφορετικές τιμές του k στην KNN παλινδρόμηση.

Method	RMSE	R^2	Time elapsed
Nearest Neighbor (k=2)	2.4677	0.8580	0.01 sec

Πίνακας 2 – Τελικό αποτέλεσμα για KNN regression.

Από τα παραπάνω δεδομένα προκύπτει ότι ο KNN regressor για k=2 είναι αρκετά αποδοτικός, με R^2 κοντά στο 0.86. Θα δούμε τώρα αν το RBF δίκτυο θα μπορέσει να περάσει τον KNN σε απόδοση.

3 Radial Basis Function ANN

Όπως αναφέρθηκε και παραπάνω, ένα RBF δίκτυο αποτελεί ένα πολύ συγκεκριμένο τύπο δικτύου, με 3 συνολικά στρώματα:

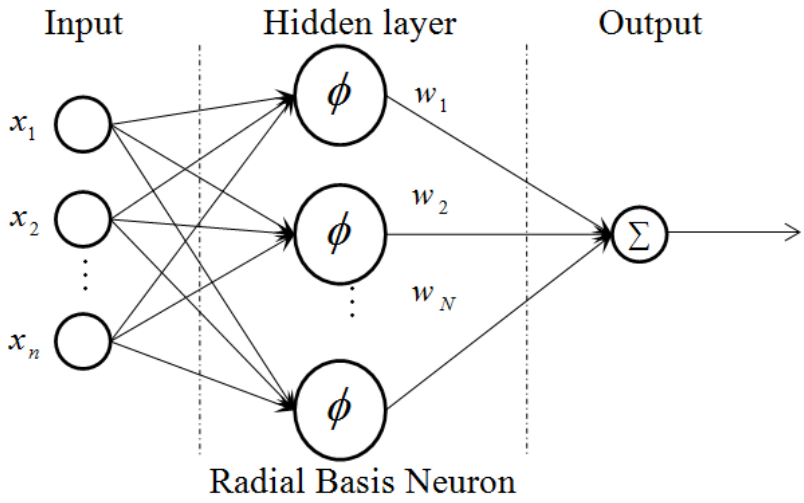
1. Το πρώτο στρώμα το οποίο λαμβάνει την είσοδο του μοντέλου.
2. Το κρυφό στρώμα το οποίο μετασχηματίζει κάθε είσοδο με μια Radial Basis Function, η οποία έχει την εξής μορφή:

$$f(x) = e^{-\beta * \|x - c_i\|^2}, \qquad \beta = \frac{1}{2\sigma_i^2}$$

όπου c_i και σ_i το κέντρο και η διασπορά κάθε νευρώνα.

3. Το στρώμα εξόδου, το οποίο εκτελεί τον κλασσικό γραμμικό ΜΣ στην έξοδο του RBF στρώματος.

Η βασική διαφορά ενός RBF δικτύου με ένα MLP είναι ότι το RBF μοντέλο έχει **δύο στάδια εκπαίδευσης**. Πρώτα υπολογίζονται τα κέντρα c και η διασπορά σ κάθε νευρώνα, και ύστερα εκπαιδεύεται το υπόλοιπο δίκτυο με τον κλασσικό BP τρόπο.



Εικόνα 1 - Η μορφή ενός απλού RBF δικτύου.

Γενικά, υπάρχει η δυνατότητα συνεχούς αλλαγής των c (μετατόπιση των Γκαουσιανών) και σ (αλλαγή της διασποράς), ώστε να εκπαιδεύονται και αυτές οι παράμετροι με το υπόλοιπο δίκτυο. Αυτό όμως αλλάζει δραματικά την συνάρτηση του νευρωνικού δικτύου, και προκαλεί μεγάλη αστάθεια. Γι' αυτό το λόγο οι δύο αυτές παράμετροι θα επιλεγούν μόνο μια φορά στην αρχή, και μετά θα **παραμένουν σταθερές**. Συγκεκριμένα, τα κέντρα θα υπολογιστούν με χρήση του αλγορίθμου **KMeans** της sklearn, και ως σ επιλέγεται η εξής τιμή για όλους τους νευρώνες:

$$\sigma = \frac{d_{max}}{\sqrt{2P}}$$

οπότε οι τιμές των "**beta**" τέθηκαν ίσες με:

$$\beta = \frac{1}{2\sigma_i^2} = \frac{1}{\frac{2d_{max}^2}{2P}} = \frac{2P}{2d_{max}^2} = \frac{P}{d_{max}^2},$$

όπου **P**: πλήθος centroid, και $\mathbf{d_{max}}$: μέγιστη απόσταση μεταξύ οποιονδήποτε 2 centroid. Η επιλογή αυτή για τις διασπορές είναι η πιο διαδεδομένη.

Η υλοποίηση του δικτύου θα γίνει μέσω της βιβλιοθήκης Keras, η οποία όμως δεν διαθέτει έτοιμο στρώμα για RBF. Γι' αυτό χρησιμοποιείται υλοποίηση από

GitHub repository[3] η οποία χτίζει το στρώμα ώστε να μπορεί να χρησιμοποιηθεί από την βιβλιοθήκη των Sequential model. Η υλοποίηση αυτή όμως έπρεπε να τροποποιηθεί για να λειτουργήσει στο παρών πρόβλημα και για να μπορεί να γίνει αργότερα το fine-tuning του δικτύου. Η αρχικοποίηση των κέντρων γίνεται μέσω της κλάσης KMeansInit, και η ανάθεση των τιμών c και betas γίνεται στην συνάρτηση build της κλάσης RBFLayer, όπου δίνεται και η δυνατότητα τα betas να ξεκινούν με την τιμή 1 και να είναι εκπαιδεύσιμα, αλλά δεν προτείνεται για το παρών πρόβλημα. Στην συνάρτηση call της ίδιας κλάσης υπολογίζονται οι αποστάσεις των δειγμάτων από τα κέντρα και προκύπτει το αποτέλεσμα της RB συνάρτησης.

Στην διαδικασία του fine-tuning που θα ακολουθήσει καθώς και στην απλή δοκιμή του μοντέλου θα χρησιμοποιηθούν οι μετρικές **RMSE** και **R^2** για την αξιολόγησή του, και η μετρική **MSE** σαν συνάρτηση κόστους. Η επιλογή του MSE έναντι του RMSE για την συνάρτηση κόστους γίνεται επειδή το τετράγωνο έχει την ιδιότητα να «μεγεθύνει» τα μεγάλα errors, οπότε «τιμωρεί» τα μοντέλα πιο πολύ για μεγάλα σφάλματα. Επίσης «τιμωρεί» τα μοντέλα μέσω της αύξησης και του μέσου σφάλματος που προκύπτει όταν χρησιμοποιείται αυτή η μετρική (MSE). Αντίστοιχα, χρησιμοποιείται η RMSE στην αξιολόγηση επειδή η ρίζα επαναφέρει το αποτέλεσμα στην αρχική του μονάδα μέτρησης (π.χ. στο παρών πρόβλημα dB αντί για dB^2). Τέλος, οι μετρικές RMSE και R^2 δείχνουν με διαφορετικό τρόπο το ίδιο αποτέλεσμα, και υπάρχει άμεση σύνδεση μεταξύ τους μέσω της MSE:

$$R^2 = 1 - \frac{MSE}{Var_{input}}$$

3.1 RBF Model Fine-Tuning

Σε αυτό το στάδιο θα προσπαθήσουμε να βρούμε τις βέλτιστες υπερπαραμέτρους του μοντέλου, για να δούμε μετά την επίδοσή του στο test set. Η επιλογή θα γίνει μέσω της βιβλιοθήκης keras-tuner, και συγκεκριμένα με τον **RandomSearch tuner**. Ο βελτιστοποιητής αυτό επιλέγεται επειδή η αρχικοποίηση των k-means clusters είναι αργή διαδικασία, ειδικά για μεγάλο πλήθος κέντρων (λόγω του d_{max} στο σ), οπότε θέλουμε ο έλεγχός μας να σταματήσει νωρίτερα, χωρίς να κοιτάξει όλο το πλήθος των συνδυασμών. Επομένως, ο RandomSearch tuner θα μας δώσει μια **υποβέλτιστη λύση**, ως αντίποινο της εξοικονόμησης χρόνου και υπολογιστικών πόρων (περιορισμοί από το Google Colab δεν επιτρέπουν κατάχρηση).

Συνήθως, το πλήθος των RBF νευρώνων και συνεπώς των κέντρων του K-Means επιλέγεται σαν ποσοστό του πλήθους των δειγμάτων στο train set, οπότε οι υπερπαράμετροι προς επιλογή και οι πιθανές τιμές τους είναι οι εξής:

[3] *Vidnerová, Petra. RBF-Keras: an RBF Layer for Keras Library. 2019. Available at* https://github.com/PetraVidnerova/rbf_keras

- rbf_neurons = [10%, 20%, 30%, 50%, 90%]
- learning_rate = [0.1, 0.01, 0.001]

Ο optimizer που επιλέγεται είναι ο SGD, και ο έλεγχος του keras-tuner γίνεται με batch_size = 15 και epochs = 100. Για την επιλογή του κατάλληλου συνδυασμού ελέγχεται η μετρική R^2 να είναι μέγιστη. Επίσης, για κάθε συνδυασμό γίνονται 5 πειράματα (trials) και λαμβάνεται ο Μ.Ο. των αποτελεσμάτων.

Η (υπο)βέλτιστη λύση που προέκυψε από τον tuner είναι:

```
Trial 13 Complete [00h 12m 16s]
val_R_squared: 0.7124560832977295

Best val_R_squared So Far: 0.7124560832977295
Total elapsed time: 02h 15m 09s
INFO:tensorflow:Oracle triggered exit

The hyperparameter search is complete. The optimal number of
neurons for the RBF layer is 503, and the optimal learning rate
is 0.1.
```

Παρατηρούμε, λοιπόν, ότι ο tuner τελείωσε στην 13η δοκιμή αντί των συνολικών 15, και διάλεξε τον συνδυασμό **[rbf_neurons, learning_rate] = [50%, 0.1]**, ο οποίος σημείωσε μέγιστο mean val_R_squared **0.7124**.

Ενδεικτικά παρουσιάζονται και τα αποτελέσματα διάφορων συνδυασμών, για να φανεί κυρίως η επίδραση του πλήθους των κέντρων στην επίδοση του μοντέλου.

Neurons	Learning Rate	RMSE	R^2
10%	0.1	4.0221	0.5740
20%	0.1	3.7220	0.6410
30%	0.1	3.3479	0.7010
50%	0.1	3.2777	**0.7124**
90%	0.1	3.2782	0.7011
50%	0.01	4.8181	0.4190

Πίνακας 3 - Μ.Ο. μετρικών για διάφορους συνδυασμούς των υπερπαραμέτρων.

Από τις παραπάνω δοκιμές προκύπτει ότι μέχρι ένα σημείο η αύξηση των νευρώνων του RBF στρώματος (άρα και τον κέντρων των cluster) αυξάνει την τιμή της μετρικής R^2, αλλά μετά από αυτό η επίδοση σταθεροποιείται ή μειώνεται μερικώς. Το αποτέλεσμα αυτό είναι λογικό, αφού όσο περισσότερο σπάει ο χώρος, τόσο καλύτερα μπορεί να προσεγγιστεί το πρόβλημα, αλλά για μεγάλο πλήθος clusters επικρατεί μεγάλη σύγχυση και τα δείγματα μπορούν να καταλήξουν εύκολα

σε λάθος (μικρή) περιοχή. Επίσης, παρατηρούμε ότι στο συγκεκριμένο πρόβλημα, το learning rate = 0.1 έφερε καλύτερα αποτελέσματα, ενώ το 0.01 αδυνατούσε να συγκλίνει σε κάποιο καλό τοπικό ελάχιστο.

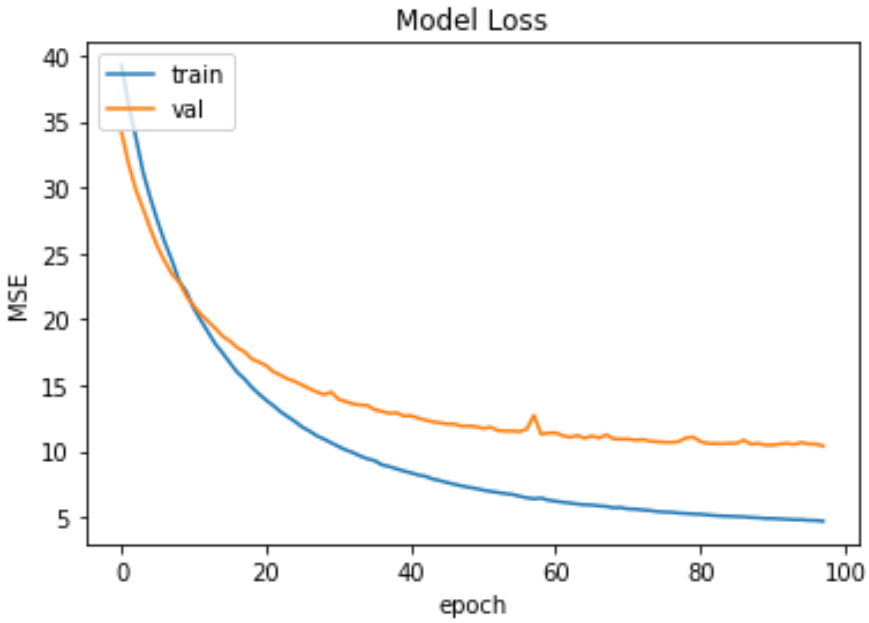
Αξίζει σε αυτό το σημείο να σημειωθεί ότι το πλήθος των κέντρων παίζει μεγάλο ρόλο στον χρόνο που θέλει το χτίσιμο του RBF στρώματος, καθώς με την διασπορά που έχει επιλεγεί, είναι αρκετά χρονοβόρο το να βρεθεί η μέγιστη απόσταση μεταξύ οποιονδήποτε 2 κέντρων ώστε αυτή να υπολογιστεί. Φυσικά η διαδικασία αυτή ανήκει στο κομμάτι του training, οπότε θα γίνεται μία μόνο φορά για συγκεκριμένο πλήθος νευρώνων. Τα τελικά κέντρα και η διασπορά τους μπορούν να εξάγονται σε αρχείο και να επαναχρησιμοποιούνται.

3.2 Fine-tuned RBF Model

Παραπάνω είδαμε ότι το μοντέλο που επιλέχθηκε έχει τις παραμέτρους [rbf_neurons, learning_rate] = [50%, 0.1]. Θα δούμε τώρα την επίδοσή του στο test σετ, και θα εξάγουμε τις train loss και validation loss καμπύλες. Τα αποτελέσματα παρουσιάζονται στον πίνακα 4 και στο διάγραμμα 1.

Test RMSE	Test R^2	Training Time	Testing Time
3.7457	0.6543	145.78 sec	0.11 sec

Πίνακας 4 - Αποτελέσματα fine-tuned RBF μοντέλου στο test set.



Διάγραμμα 1 - Loss curves του fine-tuned RBF μοντέλου.

Συμπεραίνουμε, λοιπόν, από τα παραπάνω ότι το RBF μοντέλο δεν κατάφερε να ξεπεράσει τον KNN regressor στην επίδοση, αλλά έχει ένα σχετικά ικανοποιητικό R^2 score. Ένα μεγάλο κομμάτι του training time αφορά την εύρεση του d_{max} στις διασπορές, καθώς τα 503 κέντρα είναι πολλά για προσπέλαση. Από το διάγραμμα συμπεραίνουμε ότι το μοντέλο μαθαίνει αρκετά το train set, ενώ το validation set υστερεί. Στις 100 εποχές όμως δεν προλαβαίνει να κάνει overfit και να αυξήσει το σφάλμα του validation. Η MSE καμπύλη του train φαίνεται να συνεχίζει να πέφτει, χωρίς όμως να βελτιώνεται το validation.

4 RBF With Perceptron Layer ANN

Σαν επόμενο βήμα στην έρευνά μας θα δοκιμάσουμε να συνδυάσουμε ένα RBF δίκτυο με ένα **Perceptron layer**. Γενικά, συνηθίζεται αρκετά σε τέτοια δίκτυα να προστίθεται ένα ή περισσότερα perceptron στρώματα, ώστε να εξομαλύνεται η διαχωριστική επιφάνεια που δημιουργεί το MAX του RBF δικτύου. Γι' αυτό, πριν το Dense στρώμα εξόδου θα προστεθεί ένα **επιπλέον Dense στρώμα**, με πλήθος νευρώνων που θα επιλεγεί κατά τη διαδικασία του fine-tuning.

4.1 RBF+P Model Fine-Tuning

Επειδή σε αυτό το στάδιο θα μελετήσουμε την συμπεριφορά του δικτύου μαζί με ένα στρώμα perceptron, θα αλλάξουμε τις υπερπαραμέτρους του προβλήματος και θα προστεθούν κι' άλλες. Συγκεκριμένα, θα δοκιμάσουμε διάφορα πλήθη νευρώνων για το perceptron στρώμα και θα αλλάξουν αυτές του RBF, και θα δοκιμαστούν δύο διαφορετικοί optimizers για να βρεθεί ποιος θα έχει την καλύτερη επίδοση. Αναλυτικά, οι τιμές που θα δοκιμαστούν είναι:

- rbf_neurons = [20%, 30%, 40%, 50%, 60%]
- perceptron_neurons = [128, 256, 512]
- learning_rate = [0.1, 0.01, 0.001]
- optimizer = ['sgd', 'adam']

Οι εποχές τίθενται ίσες με 200, με **early stop callback** στις 100 αν δεν βελτιώνεται η μετρική val_R^2. Επίσης, επειδή κάποιοι συνδυασμοί έφεραν μετρικές με τιμές NaN, προστίθεται **TerminateOnNaN callback** για να τις αποφύγει. Τέλος, το batch size τίθεται πάλι ίσο με 15 και τα αποτελέσματα προκύπτουν με Μ.Ο. 5 πειραμάτων ανά συνδυασμό.

Το (υπο)βέλτιστο μοντέλου που προέκυψε μέσω του RandomSearch Tuner είναι:

```
Trial 53 Complete [00h 09m 37s]
val_R_squared: 0.7192824840545654

Best val_R_squared So Far: 0.7192824840545654
Total elapsed time: 06h 26m 46s

The hyperparameter search is complete. The optimal number of
neurons for the RBF layer is 503, for the perceptron layer 256 and
the optimal learning rate is 0.001. The optimizer selected is sgd.
```

Παρατηρούμε, λοιπόν, ότι ο tuner τελείωσε στην 53η δοκιμή αντί των συνολικών 90, και διάλεξε τον συνδυασμό **[rbf_neurons, perceptron_neurons, learning_rate, optimizer] = [50%, 256, 0.001, SGD]**, ο οποίος σημείωσε μέγιστο mean val_R_squared **0.7193**. Η διαδικασία της αναζήτησης διήρκησε σχεδόν 6 ώρες και 30 λεπτά.

Αντί να παρατεθούν επιπλέον αποτελέσματα για διάφορους συνδυασμούς (κυρίως εξαιτίας του πλήθους τους), θα αναφέρουμε κάποιες παρατηρήσεις που έγιναν κατά τη διάρκεια του fine-tuning:

- Ο SGD optimizer στους περισσότερους συνδυασμούς με lr = 0.1 και 0.01 επέστρεφε σχεδόν πάντα NaN, το οποίο ενδεχομένως οφείλεται στην αργή σύγκλιση που έχει σαν optimizer, ειδικά τώρα που προστέθηκαν στο πρόβλημα περισσότερα βάρη σε σχέση με το προηγούμενο μοντέλο.
- Ο SGD φαίνεται να επιτυγχάνει περισσότερη γενίκευση σε σχέση με τον Adam, και για το μέγεθος του συγκεκριμένου dataset επιφέρει καλύτερα αποτελέσματα (πιθανώς επεξηγημένο και από το συγκεκριμένο[4] paper).
- Ο Adam με οποιονδήποτε συνδυασμό έφτανε οριακά το val_R^2 ίσο με 0.5.
- Ο Adam για μεγάλο learning rate έφερε σαν αποτέλεσμα θετικό train_R^2 και αρνητικό val_R^2.
- **Bonus:** Η χρήση dropout (0.2-0.4) μεταξύ των 2 Dense layers έφερε πάντα θετικό val_R^2 κοντά στο 0.7 και αρνητικό train_R^2. Γιατί;

Επομένως, με τα παρόντα δεδομένα, **θα επιλέγαμε το RBF+P δίκτυο** για την επίλυση του προβλήματος που παρέχει το συγκεκριμένο dataset, λόγω του υψηλότερου mean val_R^2 score που παρέχει σε σχέση με το σκέτο RBF. Απομένει να δούμε την επίδοσή του στο test set.

4.2 Fine-Tuned RBF+P Model

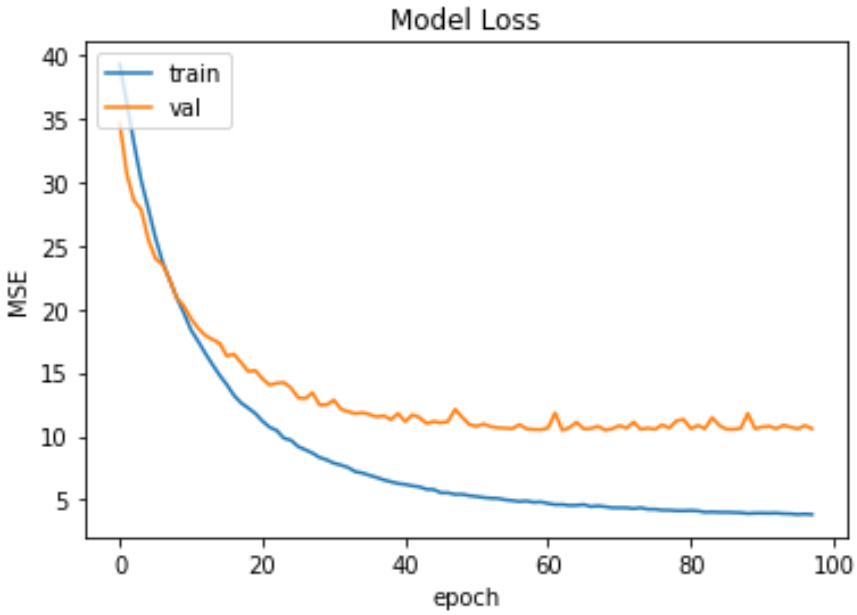
Παραπάνω είδαμε ότι το μοντέλο που επιλέχθηκε έχει τις παραμέτρους [rbf_neurons, perceptron_neurons, learning_rate, optimizer] = [50%, 256, 0.001, SGD]. Θα δούμε τώρα την επίδοσή του στο test σετ, και θα εξάγουμε τις train loss και

[4] The Marginal Value of Adaptive Gradient Methods in Machine Learning, Wilson et al. (https://arxiv.org/abs/1705.08292)

validation loss καμπύλες. Τα αποτελέσματα παρουσιάζονται στον πίνακα 5 και στο διάγραμμα 2.

Test RMSE	Test R^2	Training Time	Testing Time
3.6433	0.6718	191.61 sec	0.13 sec

Πίνακας 5 - Αποτελέσματα RBF+P δικτύου στο test set.



Διάγραμμα 2 - Loss curves του fine-tuned RBF+P μοντέλου.

Από τα παραπάνω συμπεραίνουμε ότι το μοντέλο πάλι δεν ξεπέρασε την επίδοση του KNN regressor, αλλά είχε καλύτερη επίδοση από το σκέτο RBF, έστω κατά λίγο. Οι καμπύλες σε σχέση με το προηγούμενο μοντέλο έχουν σχεδόν ίδια μορφή, αλλά εδώ βλέπουμε ότι το μοντέλο αρχίζει να κάνει overfitting, καθώς το train error πέφτει ενώ το validation error αρχίζει διστακτικά να αυξάνεται μετά τις 70 εποχές. Φυσικά ο χρόνος εκπαίδευσης είναι μεγαλύτερος σε σχέση με πριν καθώς υπάρχουν περισσότερα βάρη με την προσθήκη του perceptron layer.

5 Επίλογος

Σε αυτή την εργασία έγινε προσπάθεια επίλυσης του προβλήματος παλινδρόμησης που θέτει το Airfoil Self-Noise dataset με χρήση RBF δικτύων. Δοκιμάστηκαν διάφορες μορφολογίες, με και χωρίς χρήση perceptron στρώματος, και έγινε fine-tuning σε μεγάλο πλήθος συνδυασμών υπερπαραμέτρων. Τα δύο τελικά μοντέλα που προέκυψαν είναι:

- **RBF:** [rbf_neurons, learning_rate] = [50%, 0.1]

- **RBF+P:** [rbf_neurons, perceptron_neurons, learning_rate, optimizer] = [50%, 256, 0.001, SGD]

Τα αποτελέσματα παρουσιάζονται σε σύνοψη στον πίνακα 6.

Method	**Training time**	**Testing time**	**Test R^2 score**
KNN (2)	-	0.01 sec	**0.8580**
RBF	145.78 sec	0.11 sec	0.6543
RBF+P	191.61 sec	0.13 sec	0.6718

Πίνακας 6 - Οι μέθοδοι που δοκιμάστηκαν και τα τελικά τους αποτελέσματα.

Εύκολα προκύπτει, λοιπόν, ότι τα RBF/RBF+P δίκτυα δεν μπόρεσαν να φτάσουν σε επίδοση τον KNN, ο οποίος αν και απλός, σε τέτοιου είδους προβλήματα είναι πολύ ισχυρός. Το RBF δίκτυο με επιπλέον στρώμα perceptron έφερε καλύτερα αποτελέσματα σε σχέση με το σκέτο RBF, καθώς βοήθησε στην εξομάλυνση των διαχωριστικών επιφανειών μεταξύ των clusters.