ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

XỬ LÝ ẢNH SỐ VÀ VIDEO SỐ

Báo Cáo Bài Thực Hành – Lab 04

Giảng viên hướng dẫn Thầy Lý Quốc Ngọc

Thầy Nguyễn Mạnh Hùng

Thầy Phạm Minh Hoàng

Sinh viên thực hiện Võ Nguyễn Hoàng Kim

21127090

PHỤ LỤC

I. BẢNG TỰ ĐÁNH GIÁ 3
II. PHÂN TÍCH 3
1. Thay đổi tốc độ học – Learning rate 3
a. Khái quát 3
b. Nhận xét từ thử nghiệm 3
c. Kết luận 5
2. Thay đổi kích thước lớp – Class size 5
a. Khái quát 5
b. Nhận xét từ thử nghiệm 6
c. Kết luận 6
3. Thay đổi số lượng lớp và kiểu lớp – Number of Classes and Class Type 7
a. Khái quát 7
b. Nhận xét từ thử nghiệm 7
c. Kết luận 8
4. Thay đổi số vòng lặp – Number of Epochs 8
a. Khái quát 8
b. Nhận xét từ thử nghiệm 8
c. Kết luận 9
III. TÀI LIỆU THAM KHẢO 9

I. BẢNG TỰ ĐÁNH GIÁ

Phương pháp	Mức độ hoàn thành	Ghi chú
Thay đổi tốc độ học – learning rating	100%	-
Thay đổi kích thước lớp – class size	100%	-
Thay đổi số lượng lớp – number of classes Thay đổi kiểu lớp – class type	100%	-
Thay đổi số vòng lặp – number of epochs	100%	-

II. PHÂN TÍCH

Lưu ý:

- Mã nguồn được cài đặt dựa trên mã nguồn của giáo viên được cung cấp trên Moodle
- Trong mã nguồn này, mô hình mạng nơ-ron được thiết lập với cấu trúc gồm:
    - Lớp đầu vào (Input Layer)
    - Lớp ẩn (Hidden Layer)
    - Lớp đầu ra (Output Layer)
- Trong mã nguồn (mặc định), sẽ có:
    - Tập x_train: gồm 4 phần tử, mỗi phần tử có 2 dữ liệu đầu vào
    - Tập y_train: gồm 4 phần tử tương ứng, mỗi phần tử có 1 dữ liệu con.
- Dựa vào tập x_train, y_train, ta đưa ra các nhận xét, kết luận cho các thay đổi bên dưới

1. Thay đổi tốc độ học – Learning rate

a. Khái quát

Tốc độ học – Learning rate là một trong những tham số quan trọng để quyết định hiệu suất của mạng nơ-ron. Nó quyết định kích thước bước ở mỗi lần lặp huấn luyện trong khi hướng tới mức tối ưu của hàm mất mát. Giá trị của nó được xác định trong khoảng từ 0 đến 1 [1].

b. Nhận xét từ thử nghiệm

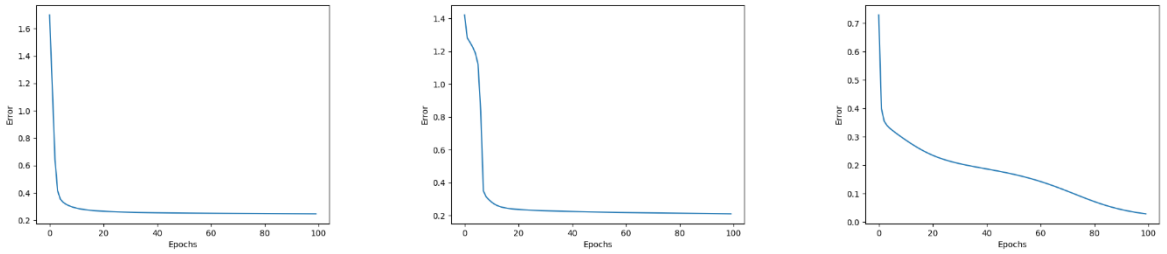
Để đưa ra nhận xét, ta thay đổi tốc độ học (learning rate) của mô hình với các mức khác nhau là 0.05, 0.1, 0.5 và 0.9. Bảng dưới đây được dựa trên kết quả trung bình sau 3 lần chạy chương trình của mỗi tốc độ

Tốc độ học	Output
0.05	[[0.0186], [0.6971], [0.7429], [0.3500]]
0.1	[[0.0533], [0.8368], [0.7842], [0.0443]]
0.5	[[0.0347], [0.9262], [0.8762], [0.2541]]
0.9	[[0.4471], [0.9953], [0.9559], [0.9622]]

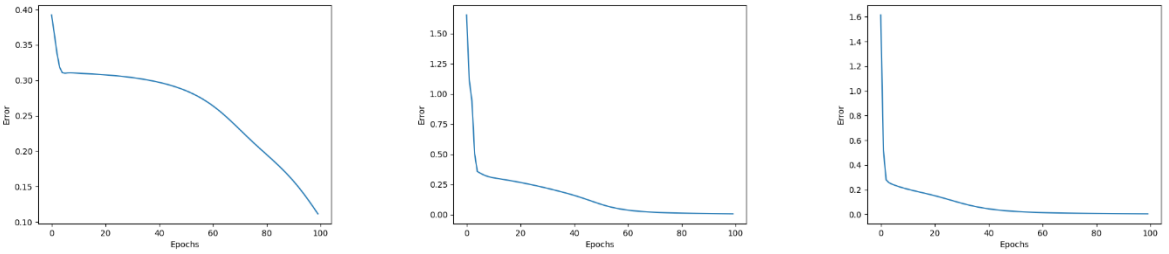
- Có thể thấy, khi tốc độ học của mô hình được tăng lên, kết quả mà nó đạt được cũng sẽ được cải thiện và tăng cao. Nhìn thấy rõ rệt nhất khi so sánh kết quả giữa tốc độ học **alpha = 0.05 và alpha = 0.9**, kết quả đầu ra được cải thiện đáng kể, Điều này cho thấy mô hình có khả năng tốt trong việc học từ tập dữ liệu huấn luyện và đưa ra dự đoán chính xác cho các mẫu trong tập này.
- Tuy nhiên, song song với kết quả tốt, việc tăng tốc độ học có khả năng khiến cho mô hình gặp vấn đề "Overfitting", vì mô hình học quá nhanh và có thể "nhớ" dữ liệu quá mức. Điều này có khả năng làm cho mô hình mất đi tính tổng quát khi gặp phải dữ liệu mới, làm ảnh hưởng đến hiệu suất của mô hình.

Với độ mất mát, từ chương trình mẫu, ta có thể quan sát tổng quát về độ mất mát tương ứng đối với từng tốc độ học thông qua nhưng biểu mẫu dưới đây:

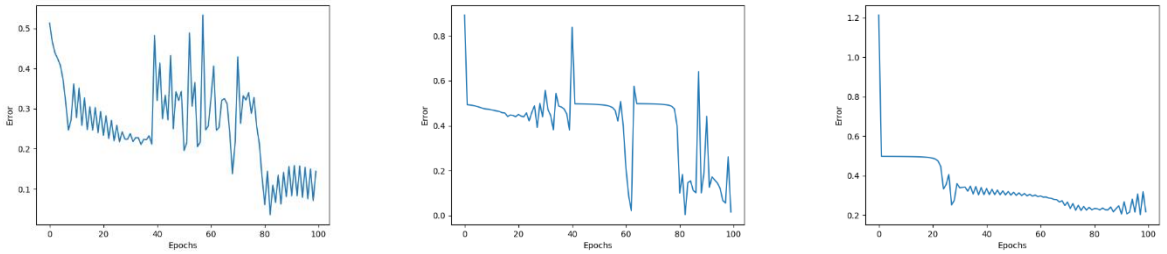
Tốc độ học 0.05

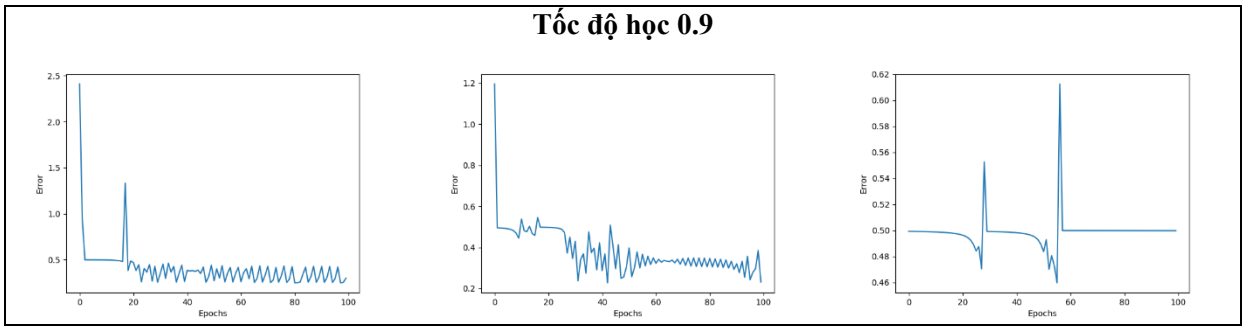


Tốc độ học 0.1



Tốc độ học 0.5





- Độ mất mát giảm dần theo từng lần huấn luyện, tuy nhiên có một số điểm đặc biệt:
    - Với tốc độ học nhỏ (**như 0.05 và 0.1**), độ mất mát được trả ra sau mỗi lần huấn luyện có sự chênh lệch khá nhỏ, gần như độ biến thiên đều nhau.
    - Tuy nhiên, với tốc độ học lớn hơn (**như 0.5 và 0.9**), có thể thấy, sự chênh lệch của các giá trị mất mát sau mỗi vòng lặp khá lớn, các giá trị biến thiên liên tục. Điều này có khả năng ảnh hưởng đến quá trình đào tạo cũng như hiệu suất của mô hình.

c. Kết luận

- Tốc độ học cao cho phép mô hình học nhanh hơn, có thể dẫn đến việc huấn luyện trở nên nhanh chóng nhưng không ổn định. Nó có thể vượt quá điểm tối thiểu do trọng số được cập nhật nhanh chóng. [1]
    - Với độ học tập quá cao có thể khiến độ mất mát biên thiên liên tục, quá trình huấn luyện trở nên không ổn định. Điều này có thể làm mất khả năng theo dõi và đánh giá hiệu suất của mô hình mạng nơ-ron.
    - Bên cạnh đó, việc đưa ra tốc độ học quá cao có thể ảnh hưởng đến kết quả học của mô hình. Kết quả được đưa ra do mô hình với tốc độ học quá nhanh sẽ không có tính tổng quát, nó chỉ tập trung dựa trên tập huấn luyện. Do đó, khả năng chính xác trên mô hình hới hoặc trên tập thử nghiệm sẽ không cao.
- Tốc độ học nhỏ cho phép mô hình học chậm và cẩn thận. Nó thực hiện những thay đổi nhỏ hơn về trọng lượng trên mỗi bản cập nhật, dẫn đến quá trình huấn luyện có thể diễn ra rất chậm và có thể rơi vào tình trạng "Hội tụ châm" hoặc bị kẹt ở một điểm cực tiểu cục bộ. [1]
    - Một tốc độ học tập hiệu quả là khi nó đủ thấp để mô hình mạng hội tụ tại một thời điểm nào đó và cũng đủ cao để mô hình huấn luyện trong một khoảng thời gian hợp lý. [2]
- Do đó, việc tìm kiếm và xác định tốc độ học tập cho mô hình là một thách thức, nó quyết định hiệu suất của mô hình rất nhiều.

2. Thay đổi kích thước lớp – Class size

a. Khái quát

Trong cấu trúc mã nguồn này, việc thay đổi kích thước lớp được cho là thay đổi tham số truyền vào trong hàm **FCLayer,** điều này tương đương với việc thay đổi kích thước của lớp ẩn. Trong phạm vi này, ta sử dụng mô hình mạng với 1 lớp ẩn để thực hiện việc thay đổi kích thước của nó.

Lớp ẩn - Hidden layer là những thành phần làm cho mạng nơ-ron trở nên "sâu" và giúp chúng có khả năng học các biểu diễn phức tạp của dữ liệu. Chúng là bộ não tính toán của các mô hình học sâu (Deep learning), cho phép mạng nơ-ron xấp xỉ các hàm và thu thập các mẫu từ dữ liệu đầu vào. [3]

Số lượng lớp ẩn và số lượng nơ-ron trong lớp ẩn xác định kiến trúc của mô hình mạng nơ-ron. Độ sâu của mạng đề cập đến số lượng lớp ẩn, ngược lại, độ rộng sẽ đề cập đến số lượng, kích thước của lớp ẩn [3]. Trong mục này, ta sẽ đề cập đến số lượng nở-ron trong lớp ẩn, tức độ rộng của lớp ẩn.

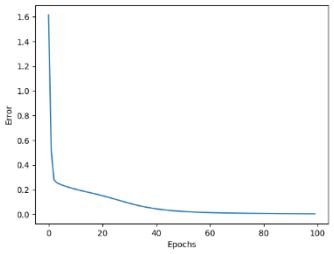
b. Nhận xét từ thử nghiệm

Bảng sau đây được lập dựa trên sự thay đổi kích thước của lớp ẩn, với tốc độ học là 0.1, giá trị được lấy là trung bình sau ba lần huấn luyện với mỗi giá trị kích thước của lớp ẩn

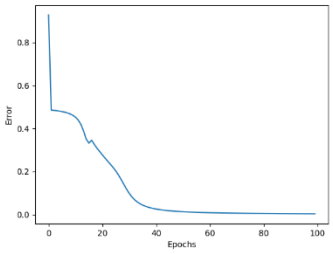
Kích thước	Kết quả đầu ra (Output)
3	[[0.0533], [0.8368], [0.7842], [0.0443]]
5	[[0.0197], [0.8100], [0.8526], [0.1909]]
8	[[0.0209], [0.9120], [0.9474], [0.1811]]

- Với kích thước càng lớn, kết quả đầu ra càng cao, dẫn đến mô hình được đào tạo gần khớp với tập dữ liệu huấn luyện được đưa ra.
- Điều này có thể được lí giải rằng: do kích thước của lớp ẩn tăng dẫn đến số lượng tham số của mô hình (trọng số (weights) và độ lệch (bias)) cũng được tăng theo (do mỗi nút trong lớp ẩn kết nối với mỗi nút trong các lớp trước và sau nó). Như đã giải thích trước đó, khi số lượng tham số càng nhiều, mô hình sẽ có khả năng học tốt hơn, đồng thời cũng tăng cao khả năng gặp phải vấn đề về "Overfitting".
- Đồng thời, trong độ mất mát sau mỗi lần lặp của chúng cũng có sự khác biệt rõ rệt, điều này gây ảnh hưởng đến hiệu suất và quá trình huấn luyện của mô hình mạng.

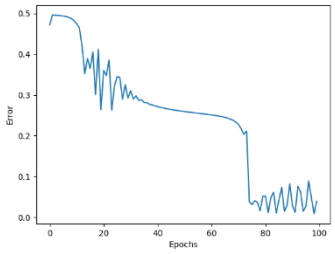
Tốc độ học 0.1 – Kích thước lớp tương ứng (size)



size = 3



size = 5



size = 8

c. Kết luận

- Các mạng rộng (lớp ẩn có kích thước lớn) giúp thu thập thông tin nhiều hơn về dữ liệu đầu vào, tuy nhiên nó có thể dẫn đến vấn đề "Overfitting". [3]
- Việc tăng nguy cơ "Overfitting" gây ảnh hưởng đến độ chính xác của mô hình, khiến nó không thể khái quát dữ liệu mới.
- Đồng thời, số lượng nút trong lớp ẩn quá lớn sẽ ảnh hưởng đến khả năng tính toán của mô hình, làm tăng đáng kể thời gian huấn luyện cũng như tài nguyên máy tính.
- Ngược lại, việc sử dụng kích thước lớp ẩn quá nhỏ sẽ dẫn đến vấn đề Underfitting (xảy ra khi dữ liệu quá ít để có thể phát hiện đủ tín hiệu trong một bộ dữ liệu phức tạp). Điều này cũng gây ảnh hưởng đến kết quả, độ chính xác cũng như hiệu suất của mô hình. [4]
- Việc tính toán để đưa ra kích thước hợp lý cho lớp ẩn là một yếu tố quan trọng, nó quyết định đến hiệu suất, độ chính xác của mô hình. Một số cách có thể được dùng để xác định kích thước cho lớp ẩn này: [4]
    - Kích thước của lớp ẩn nên nằm giữa giá trị của kích thước lớp đầu vào và lớp đầu ra.
    - Kích thước của lớp ẩn nên bằng 2/3 kích thước lớp đầu vào cộng với kích thước lớp đầu ra.
    - Kích thước của lớp ẩn nên bé hơn hai lần kích thước của lớp đầu vào.

3. Thay đổi số lượng lớp và kiểu lớp – Number of Classes and Class Type

a. Khái quát

- Số lượng lớp (Number of classes) và kiểu lớp (Class Type) thường được sử dụng trong ngữ cảnh của bài toán phân loại (Classification), trong đó:
    - Số lượng lớp: đề cập đến số lượng lớp khác nhau mà mô hình cố gắng phân loại
    - Kiểu lớp: được hiểu là loại hoặc phân loại cụ thể của từng lớp. Nó là cách mô tả chi tiết của từng lớp.
- Như vậy, trong thử nghiệm với mã nguồn này, để thay đổi số lượng lớp, ta sẽ thay đổi cả trên tập y_train để khớp với dữ liệu.

b. Nhận xét từ thử nghiệm

- Với mã nguồn ban đầu, số lượng lớp được định nghĩa là 1, đây được xem là bài toán "Binary Classification", có mô hình như sau:

```
1  # training data
2  x_train = torch.tensor([[[0, 0]], [[0, 1]], [[1, 0]], [[1, 1]]], dtype=torch.float)
3  y_train = torch.tensor([[[0]], [[1]], [[1]], [[0]]], dtype=torch.float)
4
5  # network architecture
6  net = Network()
7  net.add(FCLayer(2, 3))
8  net.add(ActivationLayer(Activation.tanh, ActivationPrime.tanh_derivative))
9  net.add(FCLayer(3, 1))
10 net.add(ActivationLayer(Activation.tanh, ActivationPrime.tanh_derivative))
```

    - Số lượng phần tử con của các phần tử trong tập y_train là 1, tương ứng với số lượng lớp được định nghĩa khi tiến hành thêm layer ở dòng 9.

```
On epoch 96 an average error = tensor(0.2002)
On epoch 97 an average error = tensor(0.1999)
On epoch 98 an average error = tensor(0.1996)
On epoch 99 an average error = tensor(0.1993)
On epoch 100 an average error = tensor(0.1991)
[tensor([[0.0347]]), tensor([[0.5778]]), tensor([[0.7537]]), tensor([[0.6267]])]
```

    - Khi đó, kết quả đầu ra của lớp Output được đưa ra như trên.

- Khi thay đổi tham số trong mã nguồn, số lượng lớp được định nghĩa sẽ là 2, khi đó, mô hình được biểu diễn như sau:

```
1  # training data
2  x_train = torch.tensor([[[0, 0]], [[0, 1]], [[1, 0]], [[1, 1]]], dtype=torch.float)
3  y_train = torch.tensor([[[0, 2]], [[1, 2]], [[2, 1]], [[0, 1]]], dtype=torch.float)
4
5  # network architecture
6  net = Network()
7  net.add(FCLayer(2, 3))
8  net.add(ActivationLayer(Activation.tanh, ActivationPrime.tanh_derivative))
9  net.add(FCLayer(3, 2))
10 net.add(ActivationLayer(Activation.tanh, ActivationPrime.tanh_derivative))
```

- Với việc thay đổi trên, số lượng phần tử con của các phần tử trong y_train được tăng lên thành 2, tương ứng với số lượng lớp được thêm vào ở dòng 9.

```
On epoch 95 an average error = tensor(0.5057)
On epoch 96 an average error = tensor(0.5056)
On epoch 97 an average error = tensor(0.5055)
On epoch 98 an average error = tensor(0.5055)
On epoch 99 an average error = tensor(0.5054)
On epoch 100 an average error = tensor(0.5054)
[tensor([[0.0137, 0.9900]]), tensor([[0.9225, 0.9980]]), tensor([[0.9911, 0.9835]]), tensor([[0.9969, 0.9927]])]
```

- Kết quả của lớp đầu ra Output như hình trên.

c. Kết luận

- Việc thiết lập số lượng lớp cũng như kiểu lớp sẽ tùy thuộc vào vấn đề mà bài toán đưa ra, cũng như phụ thuộc vào mục đích mà chúng ta muốn huấn luyện và xây dựng mô hình.

4. Thay đổi số vòng lặp – Number of Epochs

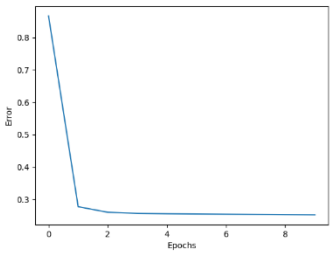
a. Khái quát

- **Vòng lặp – Epoch:** là một vòng lặp qua toàn bộ tập dữ liệu đào tạo trong quá trình huấn luyện mô hình. Mỗi epoch bao gồm việc chạy mô hình trên toàn bộ dữ liệu đào tạo một lần.
- Quá trình huấn luyện mô hình thường diễn ra qua nhiều epochs để mô hình có cơ hội học từ dữ liệu đào tạo nhiều lần và điều chỉnh các trọng số để tối ưu hóa hiệu suất. Mỗi epoch giúp mô hình "nhìn thấy" mỗi mẫu dữ liệu đào tạo và cập nhật trọng số dựa trên sự chênh lệch giữa dự đoán và nhãn thực tế [5].

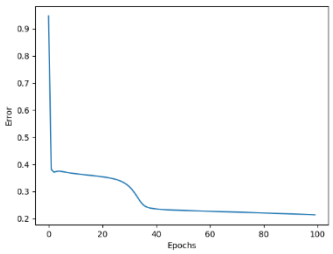
b. Nhận xét từ thử nghiệm

- Để có được nhận xét tổng quan, ta thực chạy mã nguồn với số lượng vòng lặp thay đổi khác nhau, lần lượt là 10, 100, 1000. Khi đó, ta thu được biểu đồ biểu diễn độ mất mát của từng mô hình như sau

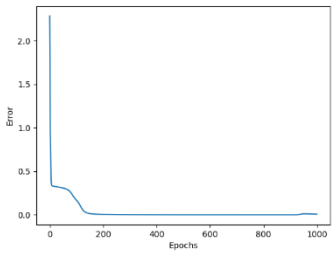
Tốc độ học 0.1 – Số vòng lặp (number)



Number = 10



Number = 100



Number = 1000

- Có thể thấy, số lần lặp – epochs có ảnh hưởng lớn đến độ mất mát của mô hình.
- Số lượng lần lặp càng lớn, mô hình mạng càng có hội duyệt qua nhiều lần để tinh chỉnh các tham số sao cho mô hình trở nên tối ưu, cũng như hiệu suất của mô hình sẽ được cải thiện.
- Đối với số lượng vòng lặp quá nhỏ (**như 10**), quá trình đào tạo của mô hình không ổn định, điều này được thể hiện qua độ mất mát của nó giảm đột ngột (giai đoạn đầu). Song song đó, với số lần lặp quá bé, có thể khiến mô hình mạng gặp phải tình trạng "Underfitting"
- Tuy nhiên trong phạm vi mã nguồn này, với mô hình khá đơn giản, việc thiết lặp số lần lặp lớn (như 1000) có thể gây ra hiện tượng "Overfitting".
- Bên cạnh đó, số lần lặp cũng gây ảnh hưởng đến thời gian huấn luyện của mô hình. Mô hình cần thời gian huấn luyện lâu cho số lượng vòng lặp lớn và ngược lại.

c. Kết luận

- Có thể thấy, việc chọn số lần lặp là một vấn đề quan trọng khi thực hiện huấn luyện mô hình.
- Việc thiết lặp số lượng vòng lặp – epochs quá mức cần thiết khiến cho mô hình dễ dàng gặp phải vấn đề "Overfitting" [6], đồng thời làm tăng thời gian huấn luyện cũng như tốn tài nguyên của máy tính.
- Tuy nhiên, nếu chọn số lượng vòng lặp – epochs quá nhỏ cũng khiến cho mô hình không học đủ từ dữ liệu đào tạo, và sẽ không thể đưa ra dự đoán chính xác trên dữ liệu mới, hay còn gọi là "Underfitting".

III. TÀI LIỆU THAM KHẢO

[1] Bhavika, "Neural Network: Introduction to Learning Rate," Study Machine Learning, [Online]. Available: https://studymachinelearning.com/neural-network-introduction-to-learning-rate/.

[2] Vinayedula, "Impact of learning rate on a model," Geeks For Geeks, 31 July 2023. [Online]. Available: https://www.geeksforgeeks.org/impact-of-learning-rate-on-a-model/.

[3] "Understanding Hidden Layers in Neural Networks," DeepAI, [Online]. Available: https://deepai.org/machine-learning-glossary-and-terms/hidden-layer-machine-learning.

[4] Jeffheaton, "Feedforward Backpropagation Neural Networks," in *Introduction to Neural Networks for Java, Second Edition* , 2008.

[5] Simplilearn, "What is Epoch in Machine Learning?," Simplilearn, [Online]. Available: https://www.simplilearn.com/tutorials/machine-learning-tutorial/what-is-epoch-in-machine-learning.

[6] https://www.geeksforgeeks.org/choose-optimal-number-of-epochs-to-train-a-neural-network-in-keras/, "Choose optimal number of epochs to train a neural network in Keras," Geeksforgeeks. [Online].