MLP 모델 성능 비교를 통한 하이퍼 파라미터 영향 평가

이동건

인하대학교 정보통신공학과

1. Introduction

딥 러닝은 이미지, 자연어, 음성 등 다양한 분야에서 높은 성능을 보이고 있다. 이러한 딥 러닝 모델들은 그 기능에 따라 여러가지 하이퍼 파라미터 (hyper parameter)를 가지며, 이러한 하이퍼 파라미터들은 모델의 성능에 큰 영향을 끼친다. 따라서 하이퍼 파라미터의 선택은 모델의 성능을 최대화하는데 결정적인 역할을 한다.

그 중에서도 다층 퍼셉트론 (이하 MLP; Multi-Layer Perceptron)[1, 2] 모델은 딥 러닝 모델들 중 가장 기초적인 인공 신경망의 한 종류로서, 최근까지도 다양한 분야에서 활발하게 활용되고 있다.[3, 4] 하지만 MLP 모델의 성능은 하이퍼 파라미터에 매우 민감하다. 이러한 하이퍼 파라미터들을 어떻게 설정하느냐에 따라 모델의 성능이 크게 달라질 수 있기 때문에, 하이퍼 파라미터 튜닝은 MLP 모델 학습에서 매우 중요한 요소이다.

본 보고서에서는 CIFAR-10[5] 데이터셋을 이용하여 MLP 모델 학습에 영향을 미치는 하이퍼 파라미터들을 분석한다. 이를 위해, 모델 은닉층의 크기 (hidden size), 배치 정규화 (Batch Normalization)[6] 사용 유무, 학습률 스케쥴러 (Learning Rate Scheduler) 등을 변화시켜가며 모델의 성능에 미치는 영향을 체계적으로 분석하고 비교한다. 이를 통해, MLP 모델에서 어떤 하이퍼 파라미터가 가장 중요한 역할을 하는지, 그리고 어떤 조합의 하이퍼 파라미터가 최적의 모델 성능을 보이는지 알아내고자 한다.

2. Backgrounds

2.1. MLP (Multi-Layer Perceptron)

인공 신경망 (Artificial Neuron Network)은 뇌의 뉴런의 구조를 모방하여 만들어진 알고리즘으로, 대표적으로 입력층과 출력층 사이에 있는 단일 계층의 뉴런, 즉 퍼셉트론 (perceptron)[7] 구조가 이에 해당한다.

퍼셉트론은 입력 데이터를 받아 내적 연산을 통해 가중치 (weight)와 곱한 뒤, 임계값과 비교하여 출력을 결정한다. 이러한 과정을 통해 퍼셉트론은 이진 분류 문제를 해결할 수 있지만, XOR 문제 같은 비선형 분류 문제를 풀 수 없다는 한계를 가진다. 이러한 한계를 해결하고자 하는 노력들이 있었고, 기존 퍼셉트론의 입력층과 출력층 사이에 여러 은닉층을 추가한 MLP [1, 2]를 통해 복잡한 비선형 문제를 해결할 수 있었다.

MLP 모델의 학습 과정은 순전파 (forward propagation), 손실 계산 (loss calculation), 역전파 (backward propagation)[8], 가중치 업데이트 (weight update)로 구성된다. 입력 데이터는 순전파를 통해 각 층에서 가중치와 활성화 함수를 이용하여 계산된 출력값이 다음 층의 입력으로 전달된다. 출력층에서는 최종 예측값이 출력된다. 손실 계산 단계에서는 출력층에서 계산된 예측값과 실제값을 비교하여 손실값을 계산한다. 이 손실값은 모델의 성능을 나타내는 지표로 사용된다. 그 다음, 역전파 단계에서는 손실값이 출력층에서 입력층으로 역으로 전파된다. 각 층에서는 출력값에 대한 손실의 미분값을 계산하고, 이를 이전 층으로 전파한다. 이 과정에서 가중치와 편향에 대한 미분값도 계산된다.

가중치 업데이트 단계에서는 각 층의 가중치와 편향이 역전파를 통해 계산된 미분값을 이용하여 갱신된다. 학습률 (learning rate)이라는 하이퍼 파라미터를 사용하여 가중치 업데이트의 크기를 조절할 수 있다.

MLP 는 위 4 단계를 반복하여 학습되며, 일정한 횟수 (epoch)나 손실값이 수렴할 때까지 계속된다. 이렇게 학습된 모델은 새로운 데이터에 대해 예측을 수행한다.

2.2. 배치 정규화 (Batch Normalization)

인공 신경망의 학습 과정 중, 모델이 학습 데이터에 과도하게 적합되어 새로운 데이터에 대한 일반화 능력이 떨어지는 경우가 발생할 수 있다. 이는 과적합 (Overfitting)이라고 부르는데, 학습 데이터가 너무 적거나 모델이 복잡한 경우에 자주 발생한다.

과적합 문제를 해결하고, 모델의 성능을 높이기 위해서 배치 정규화 (Batch Normalization) 기법[6]을 사용할 수 있다. 배치 정규화는 MLP 모델에서 사용되는 각 층의 입력 데이터에 대해 정규화 (normalization)를 수행함으로써 학습 과정에서 입력 데이터의 분포가 변화하면서 발생하는 문제들을 해결할 수 있다. 정규화 된 데이터는 평균이 0 이고 분산이 1 인 분포를 갖도록 조정된다. 이를 통해 입력 데이터의 분포가 일정하게

유지되도록 하여 학습 과정에서의 안정성을 높이고, 성능을 향상시킨다.

배치 정규화를 적용하기 위해서는 MLP 모델의 각 층마다 배치 정규화 계층을 추가해야 한다. 이 계층은 MLP 모델에서 각 층의 활성화 함수 이전에 위치하며, 입력 데이터에 대한 정규화 처리를 수행한다. 정규화 처리 이후에는 각 층의 활성화 함수를 거치게 된다.

배치 정규화를 적용하면 MLP 모델의 학습 과정에서 입력 데이터의 분포가 안정적으로 유지된다. 이를 통해 학습 과정에서의 안정성을 높이고, 그로 인해 모델의 성능을 향상시킬 수 있다.

2.3. 학습률 스케줄러 (Learning Rate Scheduler)

학습률은 가중치 업데이트 시에 얼마나 큰 갱신값을 사용할지를 결정하는 하이퍼 파라미터이다. 학습률이 너무 작으면 학습 속도가 느려지고, 학습률이 너무 크면 발산(diverge)할 가능성이 있으므로, 학습률을 조정하는 것은 인공 신경망의 성능을 개선하는 데 큰 영향을 끼친다.

학습률 스케줄러 (Learning Rate Scheduler)는 학습률을 조정하는 기법으로, 모델의 학습 과정 중에 학습률을 동적으로 변경하여 모델의 성능을 개선한다. 이를 통해 최적의 학습률을 찾아내고, 학습률이 일정한 수준으로 유지되면서 모델의 수렴을 더욱 효과적으로 돕는다.

다양한 학습률 스케줄링 방법 중 일부를 예로 들면, StepLR, ExponentialLR, LambdaLR, CosineAnnealingLR, CosineAnnealingWarmRestarts 등이 있다. 이러한 방법들은 각각 학습률을 어떻게 조절하는지에 따라 다르며, 이를 통해 모델의 성능을 개선할 수 있다. 예를 들어, StepLR 은 지정된 스텝마다 학습률을 감소시키는 방식으로 작동하고, ExponentialLR 은 지수 함수를 사용하여 학습률을 감소시킨다. LambdaLR 은 사용자 정의 함수를 통해 학습률을 조절하며, CosineAnnealingLR 은 코사인 함수를 사용하여 학습률을 조절한다. 마지막으로, CosineAnnealingWarmRestarts 는 학습률을 주기적으로 감소시키는데, 이 때 이전 주기의 학습률을 초기값으로 사용한다. 이러한 방법들을 사용하여 MLP 모델의 성능을 비교하고, 하이퍼 파라미터의 영향을 평가할 수 있다.

3. Methods

본 보고서에서는 하이퍼 파라미터가 모델의 성능에 미치는 영향을 알아보기 위하여 모델 은닉층의 크기 (hidden size), 배치 정규화의 적용, 학습률 스케줄러 등 총 3 가지 측면을 고려하였다.

3.1. 베이스라인 (Baseline) 모델

추후에 하이퍼 파라미터에 변화를 준 모델이 성능에 어느 정도의 효과를 가져오는지를 보다 명확하게 파악하기 위해 기초적인 베이스라인 (Baseline) MLP 모델을 Table 1 과 같이 정의하였다. 베이스라인 MLP 모델의 은닉층은 2 개를 사용하였으며, 두 은닉층의 크기는 256 으로 통일하였다. 입력값에 대한 출력값을 비선형적으로 만들어주기 위한 활성화 함수 (activation function)는 ReLU[9] 함수를 사용하였다.

가중치를 업데이트하는 알고리즘인 옵티마이저 (optimizer)는 Adam[10]을 사용하였으며, 옵티마이저의 학습률은 0.001 로 설정하였다. 모델의 예측값과 실제값의 차이를 나타내는 지표인 손실함수는 다중 분류 문제에 범용적으로 사용되는 손실함수인 Cross Entropy Loss 를 사용하였다.

3.2. 넓은 은닉층과 좁은 은닉층의 비교

은닉층의 크기 (hidden size)가 MLP 모델의 성능에 미치는 영향을 알아보기 위해, 은닉층의 크기를 32 부터 2,048 까지 2 배씩 늘려가면서, 총 6 개의 모델을 정의하였다. 은닉층의 개수 및 나머지 조건들과 하이퍼 파라미터들은 베이스라인 모델과 통일하였으며, 같은 MLP 모델 내부의 서로 다른 두 은닉층의 크기를 통일하였다. Table 2 를 확인하면, 은닉층의 크기가 커질 때, 모델의 파라미터 수도 함께 커지는 것을 확인할 수 있다.

3.3. 배치 정규화의 적용 여부

모델 내부에 배치 정규화 (Batch Normalization) 과정을 포함하면, 정규화 된 데이터가 과적합을 예방할 수 있다.[6]

배치 정규화는 데이터가 은닉층을 통과한 직후에 수행되며, 정규화 된 데이터는 이후 활성화 함수인 ReLU 함수를 거친다는 것을 Table 3 을 통해 확인할 수 있다.

Layer (type)	Output Shape	Param #
Linear-1	[-1, 256]	786,688
ReLU-2	[-1, 256]	0
Linear-3	[-1, 256]	65,792
ReLU-4	[-1, 256]	0
Linear-5	[-1, 10]	2,570
Total params		855,050

Table 1: 베이스라인 MLP 모델의 구조와 파라미터 수

Hidden Size	32	64	128	256	512	1024	2048
Total params	99,722	201,482	411,146	855,050	1,841,162	4,206,602	10,510,346

Table 2: 은닉층의 크기 (Hidden Size) 별 MLP 모델의 파라미터 수 (Total params)

Layer (type)	Output Shape	Param #
Linear-1	[-1, 2048]	6,293,504
BatchNorm1d-2	[-1, 2048]	4,096
ReLU-3	[-1, 2048]	0
Linear-4	[-1, 2048]	4,196,352
BatchNorm1d-5	[-1, 2048]	4,096
ReLU-6	[-1, 2048]	0
Linear-7	[-1, 10]	20,490
Total params		10,518,538

Table 3: 배치 정규화를 적용한 MLP 모델의 구조와 파라미터 수 (hidden size=2048)

3.4. 다양한 학습률 스케줄러의 비교

학습률 스케줄러를 모델에 적용할 때에는, 일반적으로는 옵티마이저 객체와 학습률 스케줄러를 함께 설정한다. 이렇게 설정된 학습률 스케줄러는 모델 학습 과정에서 매 에폭 (epoch) 마다 호출되며, 학습률 값을 업데이트한다.

본 보고서에는 고정 학습률 (0.1, 0.01, 0.001)과 StepLR, ExponentialLR, LambdaLR, CosineAnnealingLR, 그리고 CosineAnnealingWarmRestarts 등의 학습률 스케줄러를 사용한 모델을 비교한다.

4. Experiments

4.1. CIFAR-10 데이터셋

CIFAR-10[5]은 10 개의 클래스로 구성된 3x32x32 크기의 6 만 개 이미지로 이루어진 공개 데이터셋 (dataset)이다. 총 10 개의 클래스로 구성되어 있으며, Figure 1 은 그 중 일부를 나타낸 것이다. 이미지 분류를 위해 학습용 데이터셋과 테스트용 데이터셋은 각각 50,000 개와 10,000 개의 이미지로 구성하였다.

테스트용 데이터셋에 대한 모델의 성능을 평가하기 위해, 정확도 (accuracy)와 손실값 (loss)를 사용하였다. 정확도는 전체 데이터셋의 개수 중 모델이 예측한 클래스가 실제 클래스와 일치하는 데이터의 개수를 비율로 나타낸 것이며, 1 에 가까울수록 모델이 더 정확하게 예측하는 것을 의미한다. 손실값은 모델의 예측 결과와 실제 값 간의 차이를 측정하며, 값이 작을수록 모델의 예측이 실제 값과 가깝다는 것을 의미한다. 손실값이 낮을수록 모델의 예측이 더욱 정확하다고 할 수 있다.

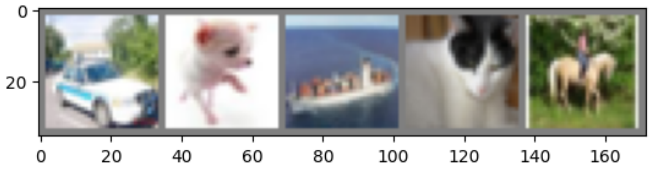

Figure 1: CIFAR-10 데이터셋의 예시.

4.2. 베이스라인 (Baseline) 모델의 성능

베이스라인 MLP 모델의 성능을 확인하기 위해, 배치 크기 (batch size)를 512 로 설정하고, 전체 에폭 (epoch)을 50 으로 설정하였다. 총 학습 시간은 9 분 59 초가 소요되었으며, Figure 2 를 통해 학습이 진행이 될 때마다 학습 세트에 대한 Cross Entropy Loss 가 감소하는 것을 확인할 수 있다. 테스트용 데이터셋에 대한 베이스라인 MLP 모델의 정확도 (accuracy)는 0.5251 를, 손실값은 0.0076 을 달성하였다.

4.3. 넓은 은닉층과 좁은 은닉층의 비교

Figure 2 는 은닉층의 크기 (hidden size)를 32 부터 2,048 까지 2 배씩 늘려가면서 정의한 총 6 개의 모델의 학습용 데이터셋에 대한 Cross Entropy Loss 를 에폭마다 나타낸 것이다. 모델의 나머지 구조와 실험 조건들은 베이스라인 모델의 성능 측정 실험과 통일하였으며, 은닉층의 크기가 커질수록 학습용 데이터셋에 대한 Cross Entropy Loss 가 작아진다는 것을 확인할 수 있다. 또한, Table 4 를 확인하면, 은닉층의 크기가 커질수록 테스트 데이터셋에 대한 정확도가 높아지는 것을 확인할 수 있다.

	Without Batch Normalization (BN)							With BN
Hidden Size	32	64	128	256	512	1024	2048	2048
Training Time	10m 1s	9m 60s	9m 58s	9m 59s	9m 59s	9m 60s	10m 1s	10m 5s
Test Accuracy	0.4847	0.4923	0.5002	0.5251	0.5434	0.5518	0.5525	**0.5911**
Test Loss	0.0031	0.0035	0.0053	0.0076	0.0081	0.0089	0.0094	0.0057

Table 4: 은닉층의 크기 (Hidden Size) 및 배치 정규화 (Batch Normalization; BN)의 적용 유무 별 MLP 모델의 성능

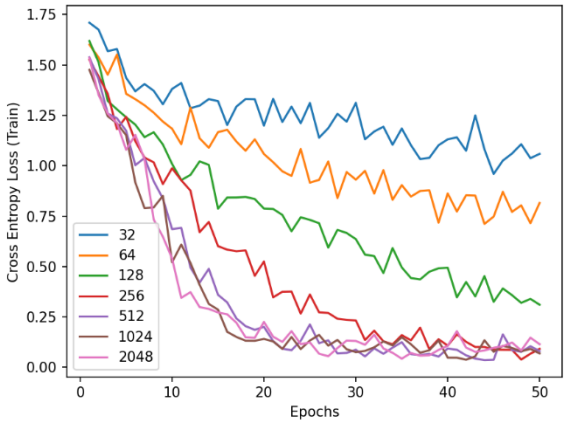

Figure 2: 은닉층의 크기에 따른 모델의 Train Loss Curve

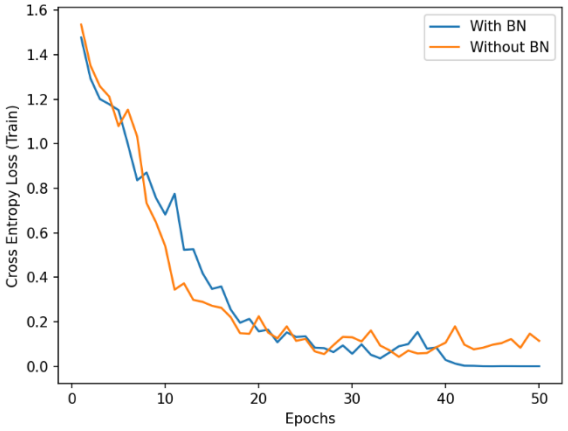

Figure 3: 배치 정규화의 적용 유무 별 Train Loss Curve

4.4. 배치 정규화의 적용 여부

앞서, 은닉층의 크기가 2,048 인 MLP 모델이 가장 높은 정확도를 가지는 것을 확인하였다. 이번 섹션에서는 해당 모델의 내부에 배치 정규화 (Batch Normalization) 과정을 추가하여 모델을 새롭게 정의하고 성능을 확인하였다.

Figure 3 에서 배치 정규화 유무의 따른 학습용 데이터셋에 대한 Cross Entropy Loss 를 확인할 수 있다. 약 40 에폭까지는 손실값이 차이가 없어 보이지만, 40 에폭 이후부터 배치 정규화를 포함한 모델의 손실값이 안정적으로 감소하는 것을 확인할 수 있다.

마찬가지로 Table 4 를 확인했을 때, 배치 정규화를 사용했을 때의 정확도가 사용하지 않았을 때의 정확도보다 대폭 상승한 것을 확인할 수 있다.

4.5. 다양한 학습률 스케줄러의 비교

이전까지 가장 성능이 좋았던 모델은 은닉층의 크기가 2,048 이고, 배치 정규화를 포함한 MLP 모델이다. 이번 섹션에서는 이 모델에 다양한 학습률 및 학습률 스케줄러를 사용하여 모델의 성능을 비교한다.

먼저, Figure 4 는 고정 학습률 및 학습률 스케줄러 별로 50 에폭 동안의 학습률의 변화와 학습용 데이터셋에 대한 Cross Entropy Loss 를 나타낸 것이다. 그리고 Table 5 는 스케줄러 별 모델의 학습 시간, 정확도, 손실값을 나타낸 것이다. 학습률을 고정시켜 모델을 학습시킨 경우에는 학습률을 0.001 로 사용한 모델의 성능이 가장 좋았다. 학습률 스케줄러를 사용한 모델들 가운데 CosineAnnealingLR 을 사용한 모델의 정확도가 가장 높았다.

LR Scheduler	Constant			StepLR	Exponential	Lambda	Cosine Annealing; CA	CA WarmRestarts
	0.1	0.01	0.001					
Training Time	10m 4s	10m 3s	10m 2s	10m 1s	10m 3s	10m 0s	10m 4s	9m 58s
Test Accuracy	0.5031	0.5631	**0.5834**	0.3478	0.2279	0.3228	0.5186	0.5141
Test Loss	0.0053	0.0069	0.0047	0.0034	0.0040	0.0036	0.0081	0.0065

Table 5: 다양한 학습률 (Learning Rate; LR) 스케줄러 별 MLP 모델의 성능

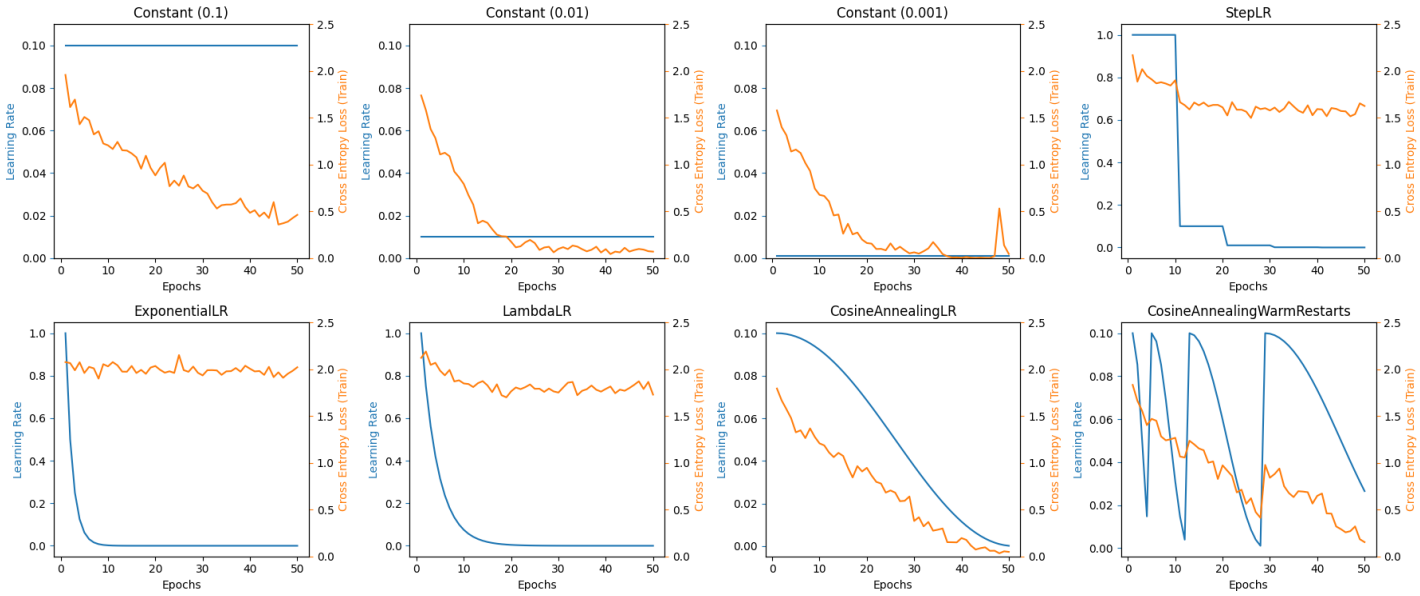

Figure 4: 고정 학습률 및 학습률 스케줄러 별 학습률의 변화와 모델의 Train Loss Curve

5. Conclusion

본 보고서에서는 현재까지 활발히 연구되고 있는 MLP 모델의 하이퍼 파라미터들을 살펴보고 다양한 조합들을 시도하며 모델의 성능을 확인하였다.

CIFAR-10 데이터셋을 통한 실험 결과, 배치 정규화를 사용하면서 넓은 은닉층 (2,048)을 가지는 모델의 정확도가 높다는 것을 확인하였고, 학습률 스케줄러보다는 고정된 작은 학습률 (0.001)로 학습을 진행할 때, 모델의 정확도가 높다는 것을 확인하였다.

학습률 스케줄러들을 사용하여 성능을 확인한 결과들 중에선 CosineAnnealingLR 을 사용했을 때의 모델의 정확도가 가장 높다는 것을 확인하였다.

6. References

[1] Collobert, R. & Benjio, S. (2004). Links between perceptrons, MLPs and SVMs. Proceedings of the Twenty-First International Conference on Machine Learning (ICML '04), (pp. 23).

[2] Alsmadi, M., Omar, K., et al. (2009). Performance Comparison of Multi-layer Perceptron (Back Propagation, Delta Rule and Perceptron) algorithms in Neural Networks. 2009 IEEE International Advance Computing Conference, (pp. 296-299).

[3] Tolstikhin, I. O., Houlsby, N., et al. (2021). MLP-Mixer: An all-MLP Architecture for Vision. Advances in Neural Information Processing Systems, 34, pp. 24261-24272.

[4] Navon, D., & Bronstein, A. M. (2022). Transformer Vs. MLP-Mixer Exponential Expressive Gap For NLP Problems. arXiv preprint arXiv:2208.08191.

[5] Krizhevsky, A. (2009). Learning Multiple Layers of Features from Tiny Images.

[6] Ioffe, S. & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37, pp. 448-456.

[7] Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review, 65 (6), pp. 386-408.

[8] LeCun, Y., Boser, B., et al. (1989). Backpropagation Applied to Handwritten Zip Code Recognition. Neural Computation 1 (4), pp. 541-551.

[9] Nair, V. & Hinton, G. E. (2010). Rectified Linear Units Improve Restricted Boltzmann Machines. Proceedings of the 27th International Conference on International Conference on Machine Learning (ICML '10), (pp. 807-814).

[10] Kingma, D. P. & Ba, J. (2014). Adam: A method for stochastic optimization. 3rd International Conference on Learning Representations, (ICLR 2015).