2024-1 인공신경망 프로젝트 최종 보고서

MNIST extended dataset 을 이용한 CNN 모델 최적화 및 분석

5 조

20192650	한수호	20180543	강구현
20192572	김민규	20192651	홍준기

https://github.com/unbroken2650/ann-project

2024.04.11 - 2024.06.11

1. 프로젝트 개요

이 프로젝트의 목표는 EMNIST 데이터셋을 활용하여 이미지 분류 작업을 수행하는 최적의 모델을 설계하는 것입니다. 이를 위해 기존 CNN 기반 모델과 자체 설계한 모델을 비교하고, 다양한 하이퍼파라미터를 조정하여 성능을 최적화했습니다. 특히, ResNet의 잔차 연결 방식을 차용한 모델을 설계하여, EMNIST 데이터셋의 복잡한 패턴을 효과적으로 학습할 수 있도록 하였습니다.

2. 수행 과정 및 역할 분담

먼저 CNN 기반의 기존 모델(이하 'baseline 모델')들과 자체적으로 설계한 모델(이하 'our 모델')을 비교하였습니다. 모델 비교 단계에서는 baseline 모델에는 LeNet5와 ResNet-50을 사용하였으며, Our 모델은 합성곱 층만을 사용한 모델을 사용하였습니다. 테스트 이후 ResNet의 잔차 연결 방식을 차용하여 새로운 모델을 설계하였습니다. 최종적으로 ResNet-50과 유사한 성능을 보였지만 훈련 시간과 파라미터 수가 훨씬 적은 두번째 모델을 프로젝트에 사용하였습니다.

선정한 모델의 옵티마이저(Optimizer), 활성화 함수, 잔차 블록(Residual Block)의 수, 학습률을 조정하는 Ablation 형태의 실험을 계획하여 진행하였습니다. 실험을 통해 가장 좋은 정확도를 보이는 하이퍼파라미터를 찾은 뒤 EMNIST byClass, byMerge, Balanced 데이터셋에 각각 학습시킨 모델을 세 가지 데이터셋에 학습하여 결과를 분석하였습니다.

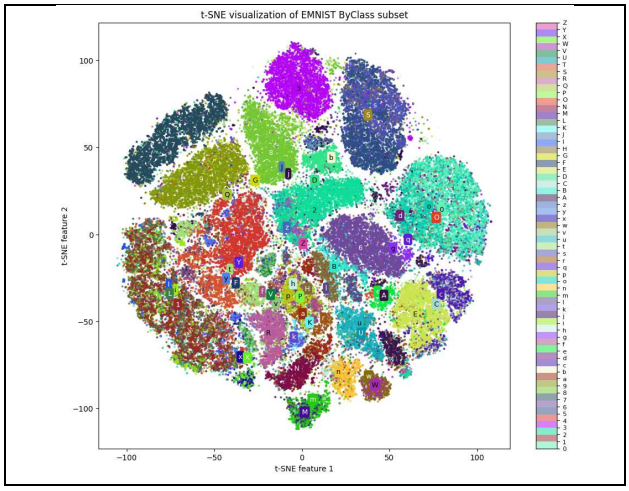
한수호	Ours 모델 설계/학습 후 분석/개선, 결과 분석, 발표 (33%)
강구현	Baseline 모델 학습 후 결과 분석, Ours 모델 설계/학습 (22%)
김민규	Baseline 모델 분석/개선, Ours 모델 분석, 보고서 작성 (22%)
홍준기	데이터셋 분석, Ours 모델 결과 분석, 보고서/발표 준비 (22%)

[표 1] 팀원 별 역할 분담

3. 데이터셋과 baseline 모델 분석

3.1. EMNIST 데이터셋

EMNIST (Extended MNIST) 데이터셋은 MNIST (Modified National Institute of Standards and Technology) 데이터셋의 확장 버전입니다. MNIST는 손으로 쓴 0~9까지의 숫자로 이루어진 데이터셋이고, EMNIST는 이를 확장하여 알파벳 소문자(a~z) 대문자(A~Z)까지 포함하는 데이터셋입니다. 이미지의 형식은 28x28픽셀 크기의 Gray scale 이미지로, 픽셀 값은 0(검은색)~255(흰색) 사이입니다. 어려운 작업을 수행하는 모델을 설계하고자 클래스의 개수와 샘플의 개수가 다른 서브셋들 중에서 가장 클래스 개수가 많고 클래스 불균형이 존재하는 byClass 데이터셋을 선택하였습니다.



[그림 1] EMNIST byClass 데이터 셋의 테스트 세트를 T-SNE(t-Distributed Stochastic Neighbor Embedding)를 이용하여 시각화한 도표

3.2. baseline 모델

LeNet은 2개의 합성곱 계층과 서브샘플링(풀링) 계층, 그리고 3개의 완전 연결 계층으로 구성된 간단한 네트워크입니다. MNIST 데이터셋과 함께 소개되었으며, 작은 이미지에 대한 효과적인 특징 추출이 가능하지만 깊은 네트워크에는 적합하지 않습니다.

ResNet은 깊은 네트워크의 학습 문제를 해결하기 위해 잔차 연결(Residual Connection)을 도입하였습니다. 이 기법은 입력 데이터를 레이어의 출력에 직접 더함으로써 깊은 네트워크에서 발생하는 그레이디언트 소실 문제를 줄였습니다. ResNet은 수십, 수백 개의 계층을 가진 깊은 네트워크에서도 성능을 유지하거나 향상시킬 수 있으며, 이미지 분류와 객체 감지 등 다양한 고해상도 이미지 처리 작업에 폭넓게 사용되었습니다.

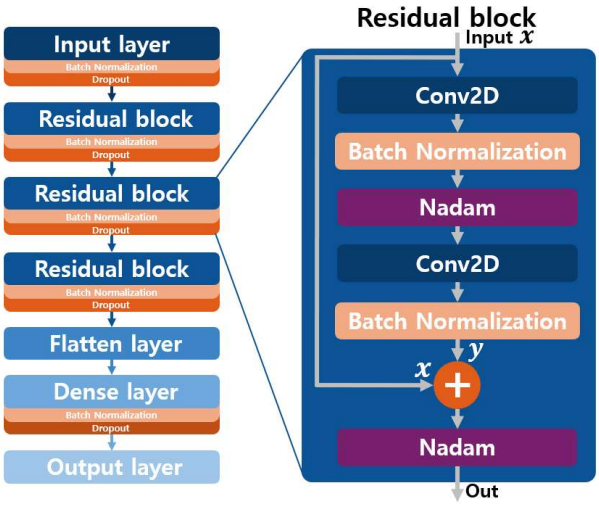
4. 프로젝트 설계 및 결과

4.1. 자체 모델 설계

처음 모델은 합성곱 계층과 최대 풀링 계층만을 사용하여 설계하였고, baseline 모델과 비교한 결과를 분석하여 잔차 연결을 사용한 모델을 새로 설계하여 개선하였습니다.

4.1.1. 잔차 연결

잔차 연결(Residual Connection)은 ResNet에서 처음 사용된 CNN을 학습시키는 기법입니다. 잔차 연결의 주 목적은 깊은 네트워크에서 발생할 수 있는 기울기 소실 문제를 해결하고, 네트워크를 효과적으로 훈련할 수 있도록 하는 것입니다. 잔차 연결의 기본 아이디어는 입력 x를 그대로 다음 층에 전달하는 것입니다. 결과적으로 output = $H(x)+x$의 형태가 되며, 여기서 $H(x)$는 CNN의 출력이고, x는 이전 층의 입력입니다.



[그림 2] Our모델의 전체 구조와 Residual block을 시각화한 순서도

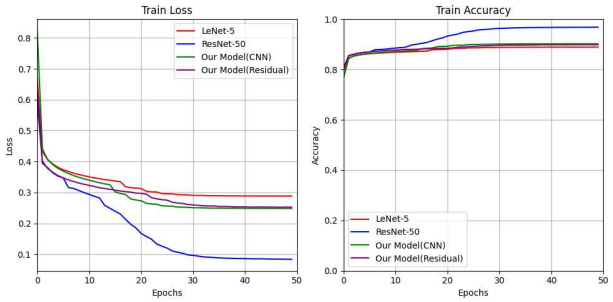
Our 모델의 전체 구조와 Residual Block을 시각화한 순서도는 [그림 1]과 같습니다. 각 Residual Block은 두 개의 CNN 레이어(Layer), 배치 정규화(Batch Normalization), Nadam활성화 함수로 구성됩니다. 이러한 Residual Block 구조를 바탕으로 Nadam 옵티마이저 와 결합하여 진행함으로써, 모델의 성능과 안정성을 더욱 향상시켰습니다.

4.1.2 Our Model

모델은 1개의 입력 층 CNN과 3개의 잔차 블록, 그리고 FC레이어로 구성되어 있습니다. ResNet과 비교했을 때, 각 계층에 Dropout을 추가하여 정규화(regularization)를 하였으며, 입력 데이터가 모델의 입력층을 통과하면 14x14 크기의 32개 피처 맵(feature map)이 생성됩니다. 이 데이터는 잔차 블록의 입력으로 들어가고, 각 잔차 블록을 통과한 후 평균 풀링 층(average pooling layer)을 지날 때마다 피처 맵의 개수는 2배로 증가하고 크기는 약 4배로 작아집니다.

4.2.baseline 모델과 Our 모델 비교

설계한 모델을 baseline 모델과 비교하였습니다. EMNIST byClass 데이터셋을 학습하여 테스트를 진행한 결과, 잔차 연결을 적용한 Our 모델이 ResNet-50보다 파라미터 개수가 적지만 가장 높은 정확도를 기록하였습니다.



[그림 3] baseline 모델과 Our 모델을 비교한 실험, 왼쪽이 training loss, 오른쪽이 train accuracy

Model	Test Loss	Test Accuracy (%)	Training Time(s)
LeNet-5	0.3671	86.85	**272.90**
ResNet50	0.7853	85.91	3086.12
WaveMix-Lite-128/7	-	**88.43**	-
Our Model(CNN)	0.4085	86.33	324.04
Our Model(Residual)	**0.3125**	**88.43**	1580.30

[표 2] baseline 모델과 Our 모델을 비교한 실험의 결과

4.3. 하이퍼파라미터 최적화 과정 및 결과 분석

EMNIST ByClass 데이터셋은 숫자와 대소문자 알파벳을 포함한 62개의 클래스로 이루어진 손글씨 데이터셋입니다. 이 데이터셋은 다양한 필기체, 복잡한 패턴, 비선형성, 노이즈 및 불균형 데이터를 포함하고 있어 최적화가 어렵습니다. 이러한 특성을 가진 데이터셋을 효과적으로 분류하기 위해, 우리는 옵티마이저, 활성화 함수, 학습률을 조정하여 성능을 비교하고 최적의 하이퍼파라미터를 찾는 과정을 진행하였습니다.

4.3.1. 옵티마이저

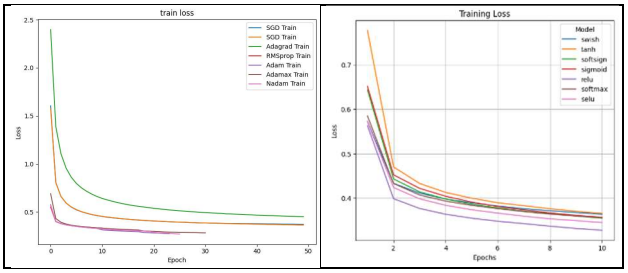
Optimizer	SGD, SGD(nestrov), Adagrad, RMSprop, Adam. Adamax, Nadam

4.2.에서 선정한 모델을 옵티마이저만 변경하여 성능을 측정하였습니다. 가장 좋은 성능을 보인 옵티마이저는 Nadam(Nesterov-accelerated Adaptive Moment Estimation)로, 88.44%의 성능을 기록하였습니다.

Optimizer	Test Loss	Test Accuracy (%)	Training Time(s)
SGD	0.3396	87.17	1277.04
SGD(nestrov)	0.3338	87.38	1286.94
Adagrad	0.3779	86.21	1289.94
RMSprop	0.3140	88.26	503.07
Adam	0.3086	88.35	605.06
Adamax	**0.3062**	88.43	775.57
Nadam	0.3082	**88.44**	784.48

[표 3] 옵티마이저에 대한 실험 결과

Nadam 옵티마이저는 Adam 옵티마이저의 적응적 학습률 조정과 모멘텀을 사용한 빠르고 안정적으로 수렴한다는 장점과 더불어 Nesterov Accelerated Gradient(NAG)를 결합하여 예측된 위치에서 기울기를 계산함으로써 더 빠르고 안정적인 수렴을 제공합니다. 이는 EMNIST ByClass 데이터셋의 다양한 클래스와 복잡한 패턴을 효과적으로 학습할 수 있게 하며, 노이즈로 인한 급격한 기울기 변화를 완화하여 안정적인 학습을 가능하게 하였습니다. 또한 Nadam은 학습률을 자동으로 조정하여 데이터의 다양한 패턴과 변화에 더 잘 대응할 수 있습니다.



[그림 4] 옵티마이저와 활성화 함수를 다르게 하여 진행한 실험, 왼쪽이 옵티마이저에 대한 실험, 오른쪽이 활성화 함수에 대한 실험

4.3.2. 활성화 함수

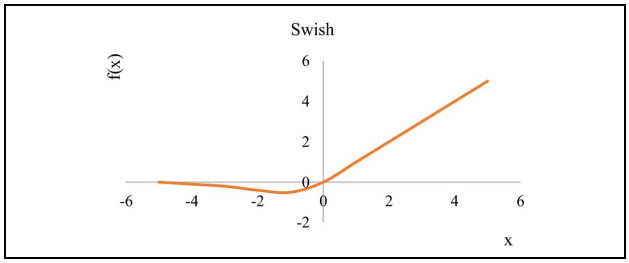
활성화 함수	ReLU, LeakyReLU, eLU, SeLU, Sigmoid, tanh, softmax, softsign, swish, GeLU

4.3.1.과 동일하게 Nadam을 적용한 뒤 활성화 함수만 변경하여 성능을 측정하였습니다. 가장 좋은 성능을 보인 활성화 함수는 Swish입니다.

Activation Function	Test Loss	Test Accuracy(%)	Training Time(s)
ReLU	0.3328	87.20	242.82
Leaky ReLU	0.3321	87.40	244.90
eLU	0.3289	87.47	246.59
SeLU	0.3384	87.24	245.21
Sigmoid	0.4001	85.20	242.76
tanh	0.3395	87.20	244.21
softmax	0.5600	83.41	273.51
softsign	0.3476	87.02	243.27
swish	**0.3163**	**87.91**	286.79
GeLU	0.3163	87.85	321.97

[표 4] 활성화 함수에 대한 실험 결과

Swish 활성화 함수는 부드러운 비선형성을 제공하여 다양한 필기체와 복잡한 글자 패턴을 더 잘 학습할 수 있으며, 입력 값이 음수일 때도 작은 출력을 유지하여 모든 입력 값을 활용할 수 있습니다. 이는 손으로 작성한 글씨에 대한 모양을 효과적으로 학습할 수 있게 하며, 그라디언트 소실 문제를 완화하여 깊은 신경망에서도 안정적으로 학습할 수 있었습니다.



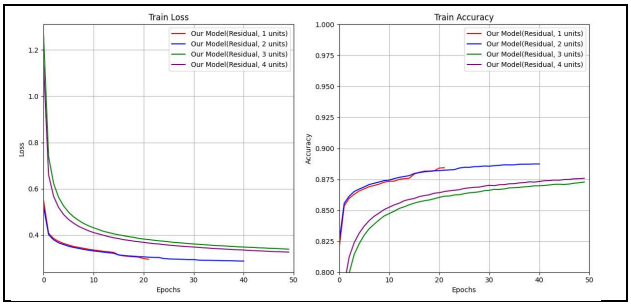
[그림 5] Swish 활성화 함수의 그래프

4.3.3. 잔차 블록의 개수

Our 모델에서 잔차 블록의 개수를 변경하여 학습을 진행하였습니다. 기존에 3개의 블록을 1, 2, 4개로 학습을 진행하고 테스트한 결과 3개의 블록이 가장 좋은 성능을 보였습니다.

# of Residual Blocks	Test Loss	Test Accuracy (%)	Training Time(s)
1 block	0.3190	88.24	1108.55
2 blocks	**0.3136**	88.37	1413.16
3 blocks	0.3264	**88.37**	1672.99
4 blocks	0.3303	88.30	1863.36

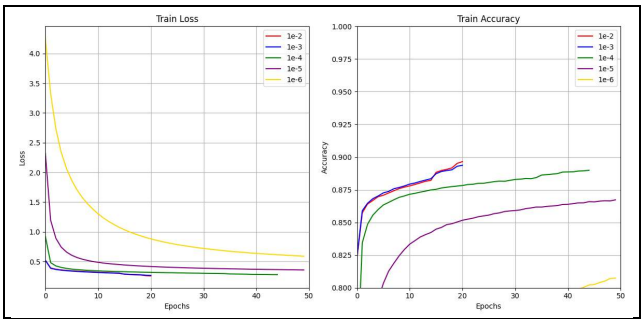
[표 5] 잔차 블록 개수에 대한 실험 결과



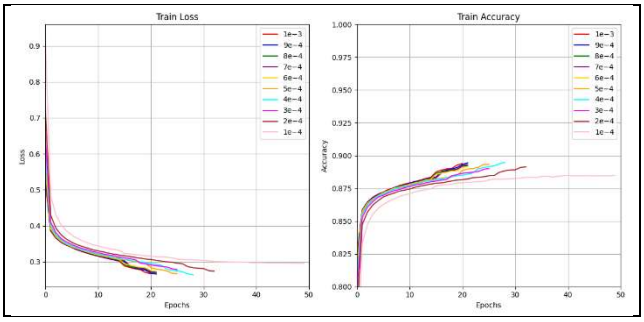
[그림 6] 잔차 블록의 개수에 대한 훈련 손실과 훈련 정확도

4.3.4. 학습률 조정

학습률을 조정하는 과정에서 여러 값들을 실험해보았습니다. 처음에는 1e-2, 1e-3, 1e-4, 1e-5, 1e-6의 범위에서 학습률을 설정하여 각각의 성능을 평가했습니다. 이 중 1e-3에서 가장 높은 정확도인 88.33%를 기록하였고, 이 결과를 바탕으로 더 세밀한 조정을 위해 1e-4와 1e-3 사이의 학습률을 10개의 값으로 나누어 추가 실험을 진행했습니다. 2e-4의 학습률이 88.41%로 가장 높은 정확도를 보였으며, 최종적으로 이 값을 모델의 학습률로 설정하였습니다.



[그림 7] 학습률에 대한 훈련 손실과 훈련 정확도(1차)



[그림 8] 학습률에 대한 훈련 손실과 훈련 정확도(2차)

Learning Rate	Test Loss	Test Accuracy(%)	Training Time(s)
1e-2	0.3223	88.02	652.85
1e-3	0.3120	**88.33**	758.32
1e-4	**0.3071**	88.32	1382.78
1e-5	0.3328	87.38	1618.95
1e-6	0.4499	84.48	1634.89

[표 6] 학습률에 대한 1차 실험 결과

Learning Rate	Test Loss	Test Accuracy(%)	Training Time(s)
1e-4	0.3071	88.32	1382.78
2e-4	**0.3067**	**88.41**	1005.71
3e-4	0.3092	88.37	932.36
4e-4	0.3090	88.26	803.46
5e-4	0.3091	88.28	709.32
6e-4	0.3109	88.37	808.81
7e-4	0.3098	88.24	673.69
8e-4	0.3106	88.26	704.26
9e-4	0.3107	88.22	669.50
1e-3	0.3100	88.22	671.94

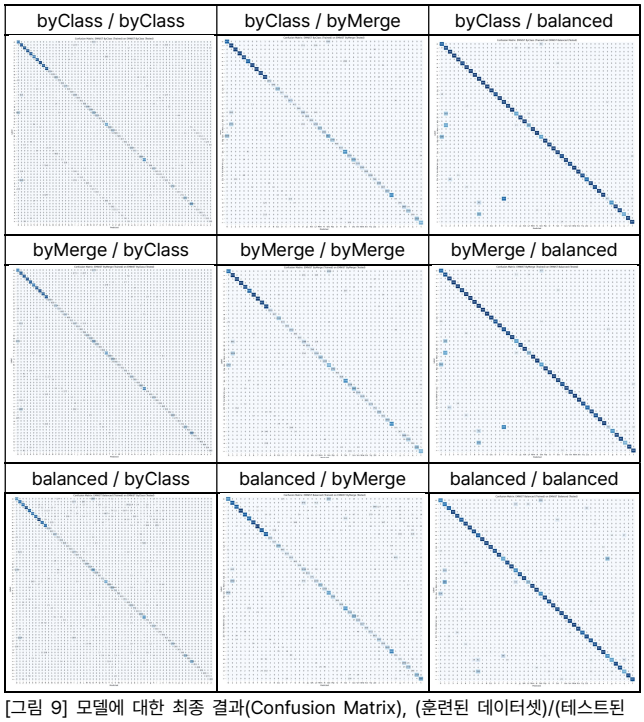
[표 7] 학습률에 대한 2차 실험 결과

4.4. 최종 결과

여러 하이퍼파라미터를 변경하며 미세 조정(fine-tuning)을 거친 모델을 EMNIST(byclass, bymerge, balanced) 데이터셋에 대하여 학습하고, 테스트 결과를 분석하였습니다.

Accuracy(%) / F1 Score / Inference Time(s)		Tested Dataset		
		byClass	byMerge	Balanced
Trained Dataset	byClass	88.27/ 0.8756/ 28.20	92.08/ 0.9167/ 22.81	90.53/ 0.9005/ 2.63
	byMerge	92.03/ 0.9150/ 22.75	91.42/ 0.9103/ 24.05	90.30/ 0.8992/ 3.46
	Balanced	86.53/ 0.8643/ 23.13	86.44/ 0.8655/ 20.53	86.99/ 0.8679/ 3.62

[표 8] 모델에 대한 최종 결과(정확도, F1 Score)



[그림 9] 모델에 대한 최종 결과(Confusion Matrix), (훈련된 데이터셋)/(테스트된 데이터셋)

5. 수행 중 문제점에 대한 분석

5.1. class_weight 적용

데이터셋의 클래스 불균형 문제를 해결하고자 TensorFlow의 class_weight를 도입하여 모델의 성능 향상을 시도하였습니다. class_weight는 각 클래스에 다른 가중치를 부여함으로써 소수 클래스의 학습을 강화하는 방식입니다. 이론적으로는 이러한 접근이 모델의 일반화 능력을 향상시킬 것으로 기대되었습니다. 그러나 실제로는 가중치 부여가 과도하게 이루어져 소수 클래스에 대한 과적합을 유발하였고, 이는 전체적인 모델 성능의 저하로 이어졌습니다. 또한, 가중치의 부적절한 설정은 학습 과정의 불안정성을 초래하였으며, 모델의 기본 구조적 한계를 가중치 조정만으로 극복하기 어려운 문제가 있었습니다. 이러한 결과들을 통해, 클래스 불균형 문제를 해결하기 위한 접근으로서 class_weight의 단순 적용만으로는 부족하며, 모델 구조의 개선, 학습 알고리즘의 최적화, 그리고 데이터 전처리 방법 등 여러가지 개선이 필요하다는 것을 알게 되었습니다.

5.2. focal loss 적용

EMNIST 데이터셋을 분류하는 성능을 향상시키기 위해 손실 함수를 크로스 엔트로피 손실 함수에서 focal loss로 실험적으로 도입하였습니다. focal loss는 클래스 불균형을 해결하고 어려운 예제에 더 많은 가중치를 부여하는 장점을 가지고 있지만, 실험 결과는 크로스 엔트로피를 사용했을 때보다 성능이 개선되지 않았습니다. 이는 모델이나 하이퍼파라미터 설정이 focal loss의 장점을 충분히 활용하기에 적합하지 않았을 가능성이 있습니다. 또한 추가적인 파라미터 조정이 이루어졌으나, 이 데이터셋과 모델 조합에서 focal loss의 효과를 극대화하기 위한 최적의 조건을 찾는 데는 한계가 있었습니다. 따라서 focal loss의 효과는 특정 상황에서 더 잘 나타날 수 있으며 일반적인 분류 상황에서는 크로스 엔트로피 손실함수가 더 효과적임을 학습하였습니다.

6. 프로젝트를 통해 얻게 된 점

모델을 직접 설계하고 개선하는 과정을 통해 수업에서 배운 지식을 더 깊게 이해하게 되었습니다. 특히, 잔차 연결을 활용한 모델을 학습률과 잔차 블록의 개수를 최적화하는 등 하이퍼파라미터를 조정하여 모델의 성능을 크게 개선하였습니다. 그리고 데이터의 불균형과 복잡성을 다루는 데 있어서 새로운 정보들을 얻을 수 있었습니다. 특정 클래스가 과도하게 많거나 적은 데이터셋에서의 학습이 어려운 부분이 많은 것을 깨달았습니다.

또한, 프로젝트를 진행하는 데에는 다양한 시행착오도 경험하였습니다. 문제를 해결하기 위해 시도했던 class_weight와 focal loss 기법이 예상과 달리 원하는 성능을 보여주지 못하는 결과가 도출되었습니다. 이러한 경험을 통해, 향후 프로젝트에서는 데이터 전처리 단계에서의 샘플링 기법이나 추가적인 정규화 기법을 도입하는 등의 전략을 더 적극적으로 고려하여 데이터 불균형 문제를 보다 효과적으로 해결할 예정입니다.

프로젝트는 기술적 성취뿐만 아니라 팀워크와 협력을 통해 기본적인 인공신경망의 개념을 이해하고 여러 가지 기법들에 대한 깊고 넓은 이해를 가질 수 있게 되었습니다.

7. 결론

이 프로젝트는 자체 설계한 모델과 기존 CNN 기반 모델의 정확도 및 추론 시간을 비교 분석하여 최적의 모델을 선정하고, 이를 EMNIST 데이터셋에 학습시켜 이미지 분류(Image Classification) 작업을 수행하는 것을 목표로 진행되었습니다. ResNet의 잔차 연결 방식을 차용한 모델을 설계하였고, 다양한 하이퍼파라미터를 변경하여 모델을 미세 조정 (fine-tuning)하였습니다. 그 결과, EMNIST byClass 데이터셋에 대해 88.27%의 정확도, 28.20초의 추론 시간을 보였습니다.

8. 참고 문헌

MNIST: Cohen, G., Afshar, S., Tapson, J., & Van Schaik, A. (2017, May). EMNIST: Extending MNIST to handwritten letters. In 2017 international joint conference on neural networks (IJCNN) (pp. 2921-2926). IEEE.

LeNet: LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

ResNet: He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

ResNet50 : Zhang, L., Bian, Y., Jiang, P., & Zhang, F. (2023). A transfer residual neural network based on ResNet-50 for detection of steel surface defects. Applied Sciences, 13(9), 5260.

Convolution Layer: O'shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. arXiv preprint arXiv:1511.08458.

Wavemix: Jeevan, P., & Sethi, A. (2022). Wavemix: resource-efficient token mixing for images. arXiv preprint arXiv:2203.03689.

Swish: Fatima, A., & Pethe, A. (2022). Periodic analysis of resistive random access memory (RRAM)-based swish activation function. SN Computer Science, 3(3), 202.