

# 중간 발표

7조

박태현, 송준규, 양은주, 정하연

## ■ 목차

1. Discover and Visualize the data
2. Prepare the data
3. Baseline Training  
: LeNet-5, ResNet-50
4. Select the model  
: 모델 선정, 추가 모델 선정
5. Plan

역할 분담

박태현	송준규	양은주	정하연
다양한 CNN 모델 분석 최종 모델 선정 평가 지표 조사	데이터셋 분석 및 전처리 최종 모델 튜닝 중간발표자 1	LeNet5 및 ResNet 구조 평가 지표 조사 최종 발표자 1	팀장 실험 및 분석 총괄 중간발표자 2 최종발표자 2

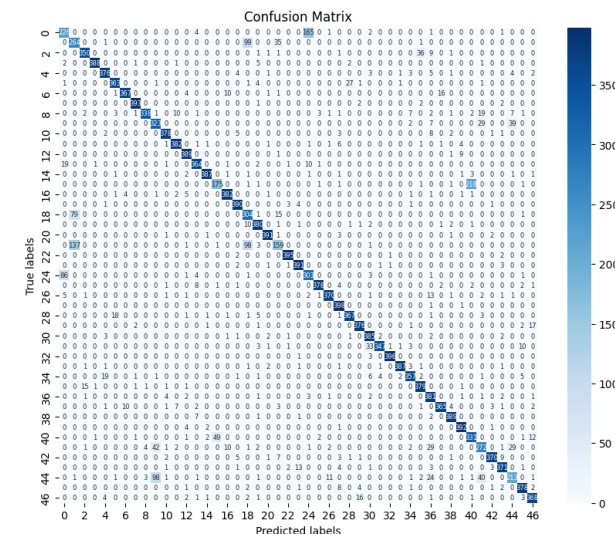
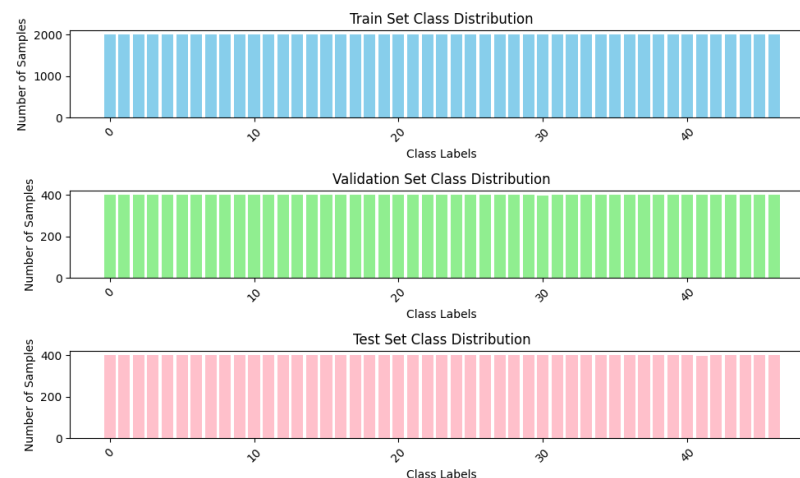
## Discover and Visualize the Data

- EMNIST 논문을 읽으며 정확한 정보를 파악함
- Kaggle로부터 EMNIST를 다운로드
- 6가지 종류의 데이터셋을 Class별로 모두 plot 후 관찰



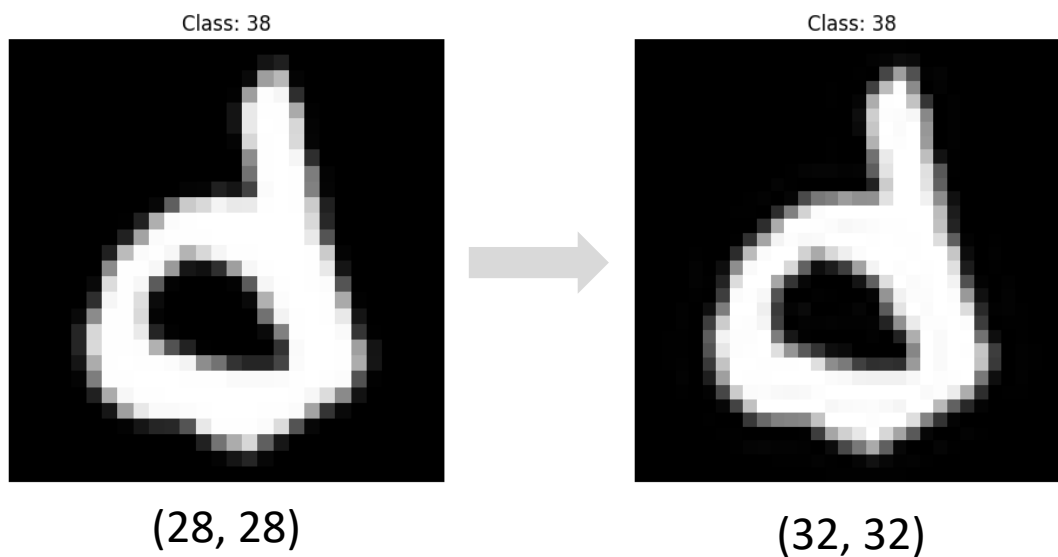
# Discover and Visualize the Data

- EMNIST Balanced Dataset
  - Train: (94000, 28, 28)
  - Valid: (18799, 28, 28)
  - Test: (18799, 28, 28)
- 클래스 별 데이터 개수의 불균형에서 오는 성능 저하를 예방하기 위함
  - 모델 성능 개선 후 시간이 있다면, Confusion matrix를 참고하여 혼동이 있는 클래스들의 데이터셋을 보완할 예정

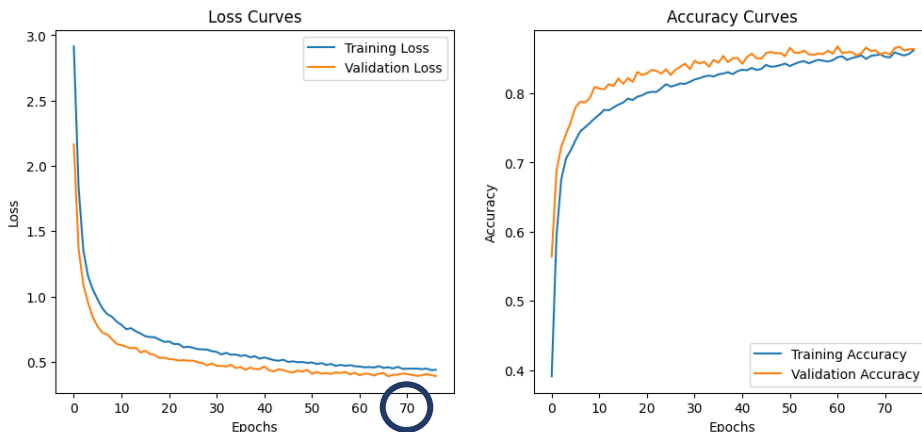
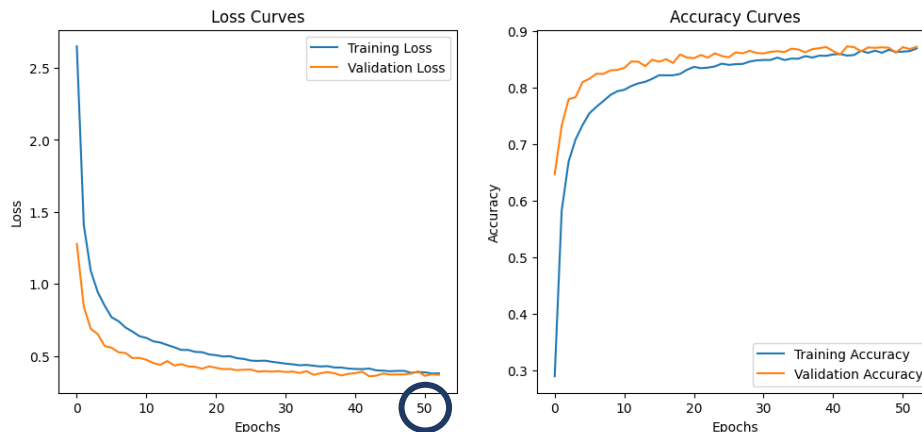


## Prepare the data

- 모델마다 최소 input\_size가 다름
  - resize\_shape=(WIDTH, HEIGHT)를 지정하여 데이터셋을 준비할 수 있는 prepare\_datasets 함수 정의.
- Ex. ResNet-50의 input\_size는 (32, 32)



# Baseline Training (LeNet-5)

	LeNet-5-old	LeNet-5-new
Activation Function	tanh, RBF	ReLU, Softmax
Learning Curve		
Training Time	3.5 sec/epoch	3 sec/epoch
	266.94 sec (4m 27.0sec)	171.13 sec (2m 51.1sec)
Test Accuracy	86.10 %	86.90 %

## Baseline Training (ResNet-50)

	ResNet-50 (default)
Learning Curve	<div><div>Loss Curves</div><div>Accuracy Curves</div><div>Loss Curves</div><div>Accuracy Curves</div></div>
Training Time	21 sec/epoch
	1966.95 sec (32m 49.3sec)
Test Accuracy	87.90 %

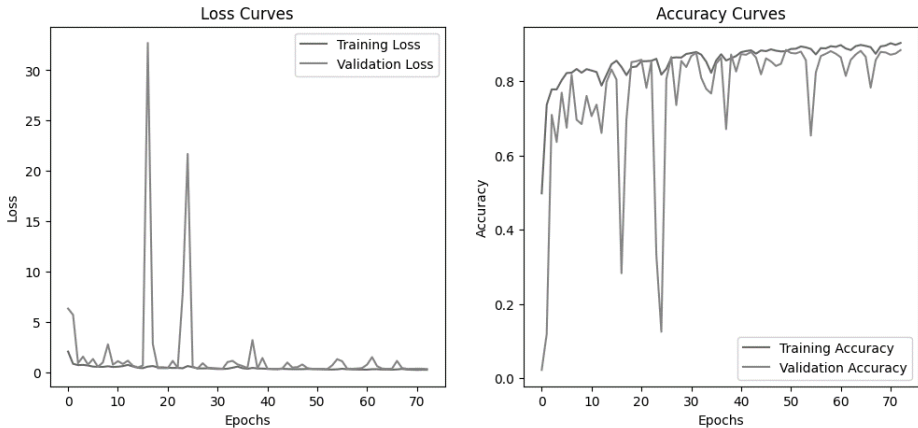
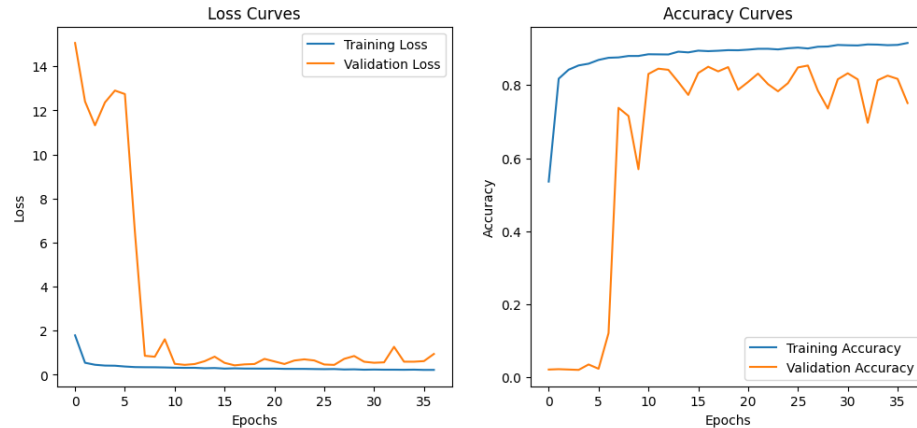


## ■ Baseline Training (ResNet-50)

1. 배치 사이즈 변경
2. Learning Rate Scheduler
  - initial\_learning\_rate=0.0005
  - ReduceLROnPlateau
3. 활성화 함수 변경
  - LeakyReLU
4. regularization 추가
  - dropout
5. 데이터 증강(data augmentation)

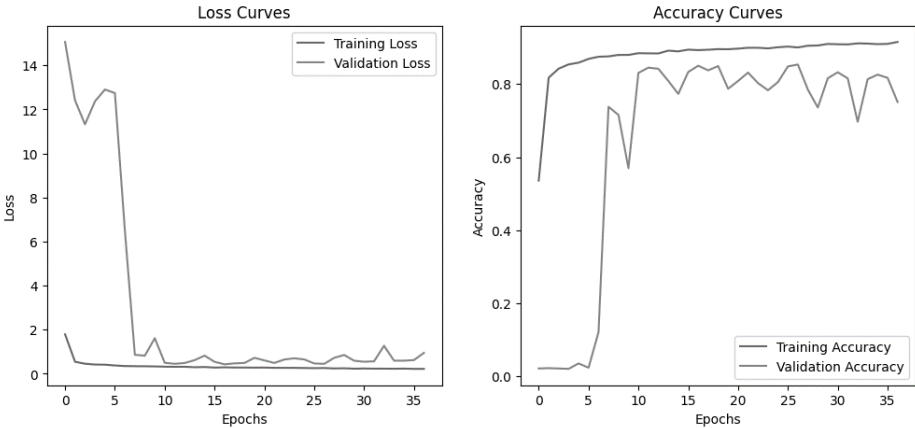
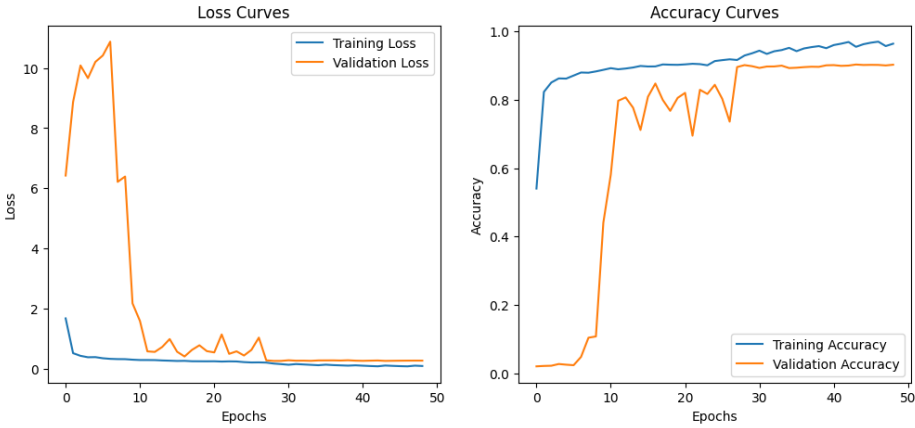
# Baseline Training (ResNet-50)

## 1. 배치 사이즈 변경 : 256 → 2048

	ResNet-50 (default)	ResNet-50 ( <b>batch_size=2048</b> )
Learning Curve		
Training Time	21 sec/epoch	20 sec/epoch
	1966.95 sec (32m 49.3sec)	1017.88 sec (17m 0.0sec)
Test Accuracy	87.90 %	76.00 %

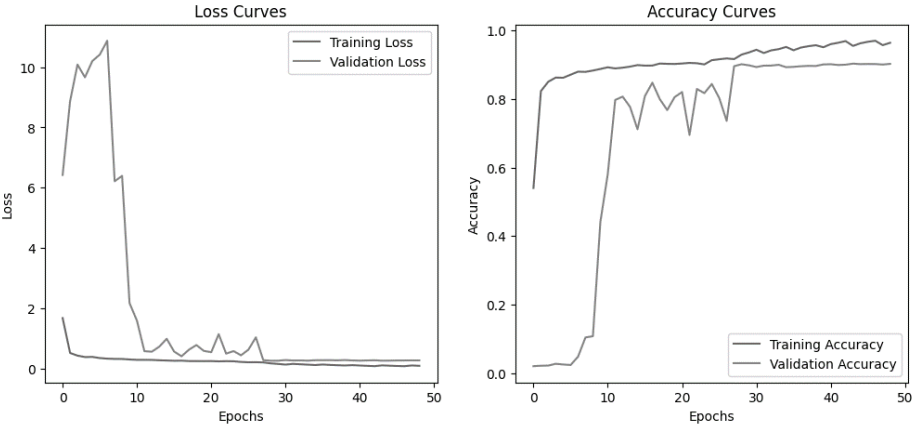
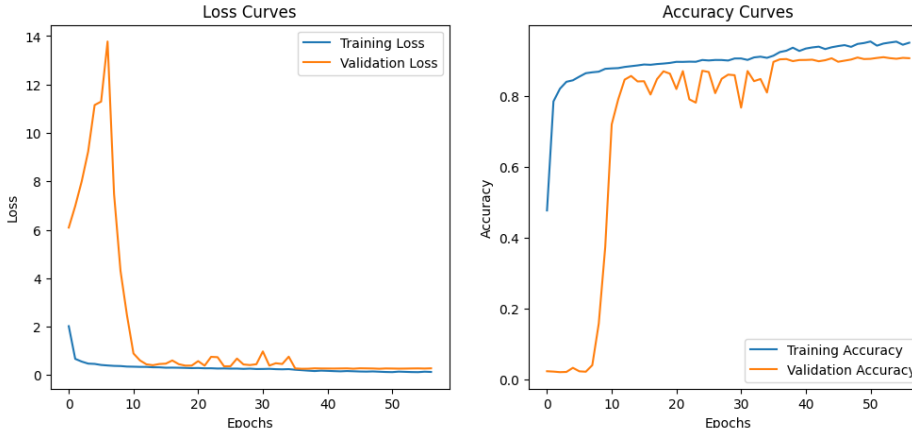
# Baseline Training (ResNet-50)

## 2. Learning Rate Scheduler

	ResNet-50 (batch_size=2048)	ResNet-50 (initial=0.0005, ReduceLROnPlateau)
Learning Curve		
Training Time	20 sec/epoch	20 sec/epoch
	1017.88 sec (17m 0.0sec)	1256.04 sec (20m 58.1sec)
Test Accuracy	76.00 %	90.60 %

# Baseline Training (ResNet-50)

## 3. 활성화 함수 변경

	ResNet-50 (initial=0.0005, ReduceLROnPlateau)	ResNet-50 (LeakyReLU)
Learning Curve		
Training Time	20 sec/epoch 1256.04 sec (20m 58.1sec)	20 sec/epoch 1522.43 sec (25m 24.5sec)
Test Accuracy	90.60 % (test_loss=0.309)	90.60 % (test_loss=0.326)

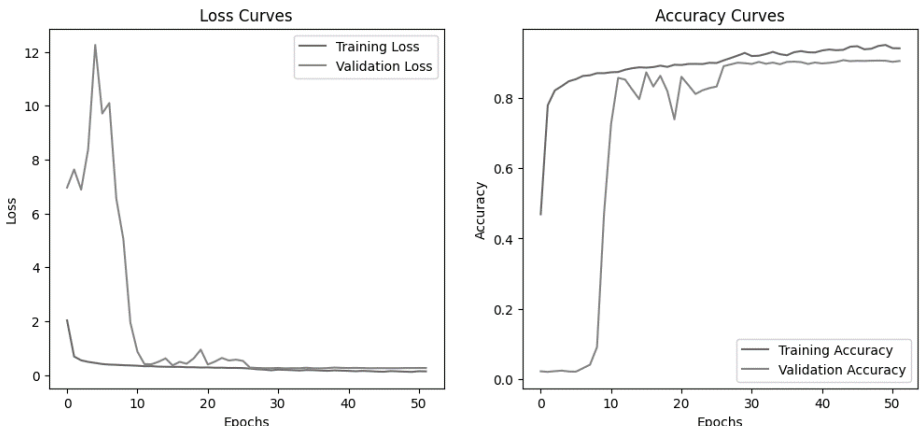
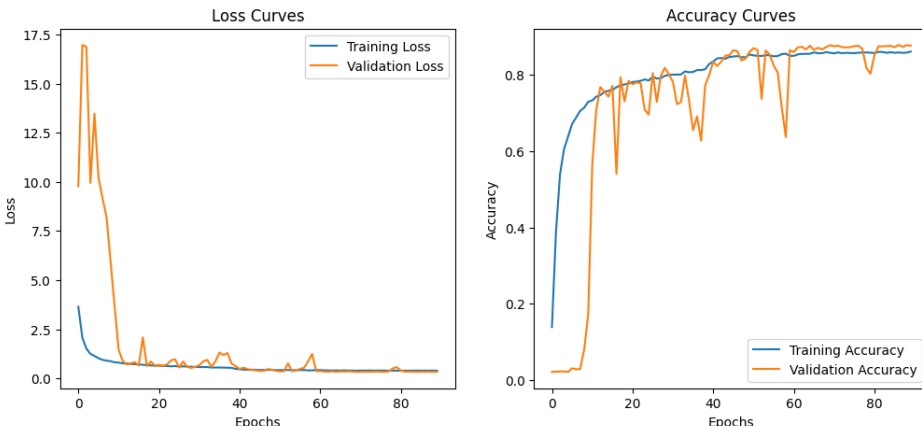
# Baseline Training (ResNet-50)

## 4. regularization 추가 : dropout

	ResNet-50 (LeakyReLU)	ResNet-50 (dropout)
Learning Curve	<p>The Learning Curve for ResNet-50 (LeakyReLU) consists of two subplots. The 'Loss Curves' plot shows Training Loss (black line) and Validation Loss (grey line) over 50 epochs. Training Loss starts at approximately 14, drops sharply to near 0 by epoch 10, and remains stable. Validation Loss starts at approximately 2, peaks at epoch 5, and then stabilizes around 0.3. The 'Accuracy Curves' plot shows Training Accuracy (black line) and Validation Accuracy (grey line) over 50 epochs. Training Accuracy starts at 0.0 and rises to approximately 0.9. Validation Accuracy starts at 0.0, rises to approximately 0.85 by epoch 10, and then fluctuates between 0.8 and 0.9.</p>	<p>The Learning Curve for ResNet-50 (dropout) consists of two subplots. The 'Loss Curves' plot shows Training Loss (blue line) and Validation Loss (orange line) over 50 epochs. Training Loss starts at approximately 12, drops sharply to near 0 by epoch 10, and remains stable. Validation Loss starts at approximately 7, peaks at epoch 5, and then stabilizes around 0.3. The 'Accuracy Curves' plot shows Training Accuracy (blue line) and Validation Accuracy (orange line) over 50 epochs. Training Accuracy starts at 0.0 and rises to approximately 0.9. Validation Accuracy starts at 0.0, rises to approximately 0.85 by epoch 10, and then fluctuates between 0.8 and 0.9.</p>
Training Time	20 sec/epoch	20 sec/epoch
	1522.43 sec (25m 24.5sec)	1449.46 sec (24m 11.6sec)
Test Accuracy	90.60 % (test_loss=0.326)	90.40 % (test_loss=0.308)

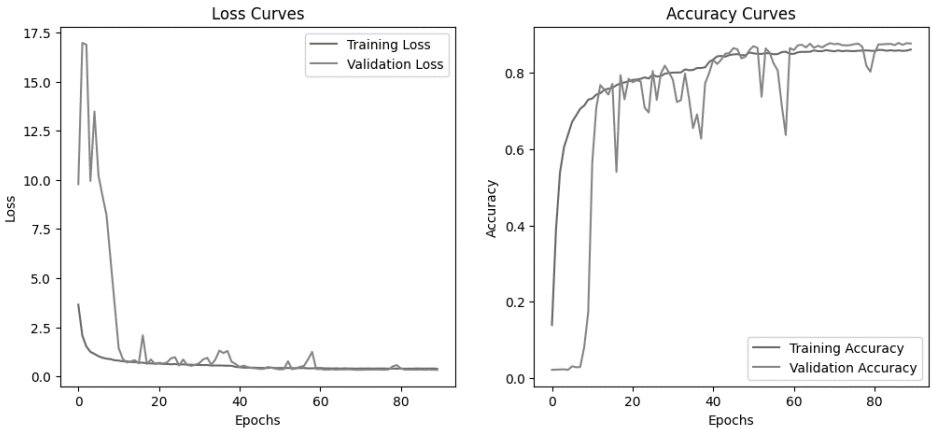
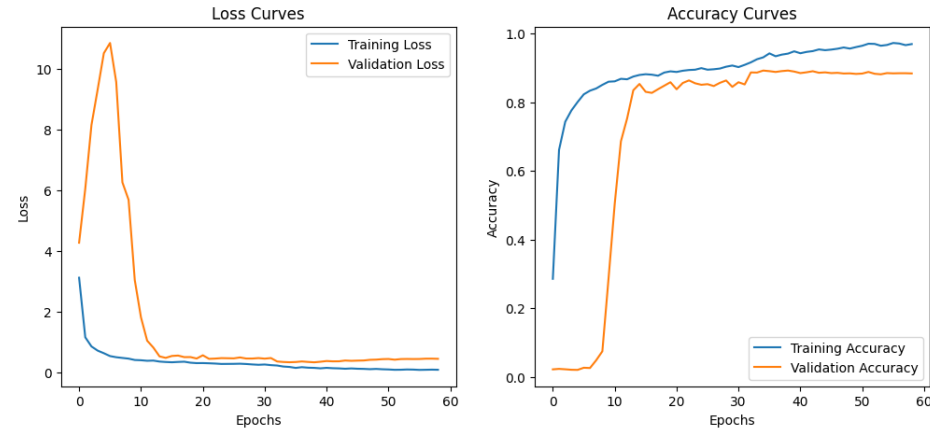
# Baseline Training (ResNet-50)

## 5. 데이터 증강(data augmentation)

	ResNet-50 (dropout)	ResNet-50 (data augmentation)
Learning Curve	 <p>The figure contains two subplots. The left subplot, titled 'Loss Curves', shows Training Loss (black line) and Validation Loss (grey line) over 50 epochs. Training loss starts at approximately 12, peaks at 13 around epoch 5, and then drops sharply to near 0 by epoch 10. Validation loss starts at approximately 7, peaks at 11 around epoch 5, and then drops to approximately 0.3 by epoch 10. The right subplot, titled 'Accuracy Curves', shows Training Accuracy (black line) and Validation Accuracy (grey line) over 50 epochs. Training accuracy starts at 0.0 and rises to approximately 0.9 by epoch 10. Validation accuracy starts at 0.0 and rises to approximately 0.9 by epoch 10.</p>	 <p>The figure contains two subplots. The left subplot, titled 'Loss Curves', shows Training Loss (blue line) and Validation Loss (orange line) over 80 epochs. Training loss starts at approximately 17.5, drops sharply to near 0 by epoch 10, and remains low. Validation loss starts at approximately 17.5, drops sharply to near 0 by epoch 10, and remains low. The right subplot, titled 'Accuracy Curves', shows Training Accuracy (blue line) and Validation Accuracy (orange line) over 80 epochs. Training accuracy starts at 0.0 and rises to approximately 0.9 by epoch 10. Validation accuracy starts at 0.0 and rises to approximately 0.9 by epoch 10.</p>
Training Time	20 sec/epoch	23 sec/epoch
	1449.46 sec (24m 11.6sec)	2728.71 sec (45m 30.9 sec)
Test Accuracy	90.40 % (test_loss=0.308)	88.40 %

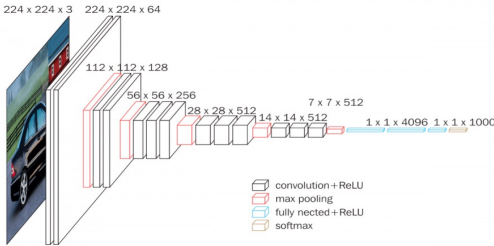
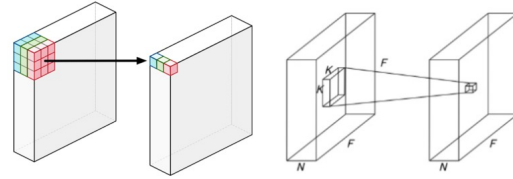
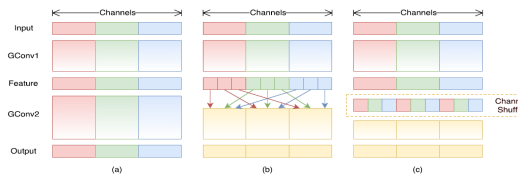
# Baseline Training (ResNet-50)

+ Zero\_padding을 추가하는 레이어로 수정

	ResNet-50 (data augmentation)	ResNet-50 (zero_padding)
Learning Curve	 <p>The figure contains two subplots. The left subplot, titled 'Loss Curves', shows Training Loss (black line) and Validation Loss (grey line) over 80 epochs. The Training Loss starts at approximately 16.5 and drops sharply to near 0 by epoch 10, remaining stable thereafter. The Validation Loss starts at approximately 1.5 and drops to near 0 by epoch 10, remaining stable. The right subplot, titled 'Accuracy Curves', shows Training Accuracy (black line) and Validation Accuracy (grey line) over 80 epochs. Both start at 0.0 and rise sharply, reaching approximately 0.85 by epoch 10 and stabilizing around 0.90.</p>	 <p>The figure contains two subplots. The left subplot, titled 'Loss Curves', shows Training Loss (blue line) and Validation Loss (orange line) over 60 epochs. The Training Loss starts at approximately 3.5 and drops to near 0 by epoch 10, remaining stable. The Validation Loss starts at approximately 4.5, peaks at about 10.5 around epoch 5, and then drops to near 0 by epoch 10, remaining stable. The right subplot, titled 'Accuracy Curves', shows Training Accuracy (blue line) and Validation Accuracy (orange line) over 60 epochs. Both start at 0.0 and rise sharply, reaching approximately 0.85 by epoch 10 and stabilizing around 0.90.</p>
Training Time	23 sec/epoch	6 sec/epoch
	2728.71 sec (45m 30.9 sec)	736.672 sec (12m 18.7 sec)
Test Accuracy	88.4 %	88.70 %

## Select the Model (모델 후보)

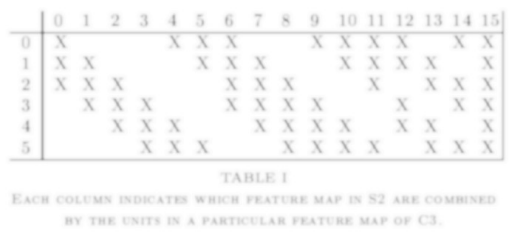
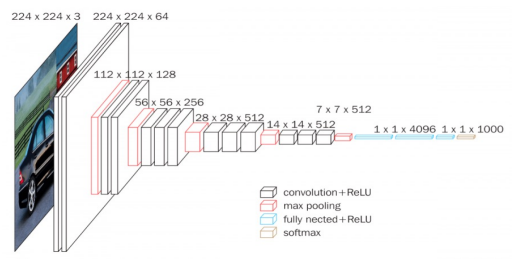
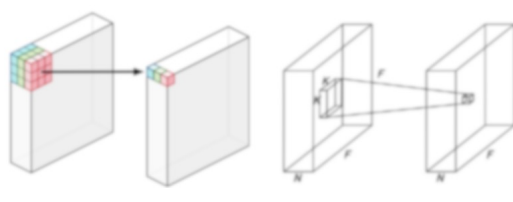

- EMNIST 데이터셋의 특성, 모델 구조 및 특성을 함께 고려하여 모델 후보 4개 선정

LeNet-5	VGG-16	MobileNet	ShuffleNet																																																																																																																													
<table border="1"><tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td></tr><tr><td>0</td><td>X</td><td></td><td></td><td></td><td></td><td>X</td><td>X</td><td>X</td><td></td><td></td><td>X</td><td>X</td><td>X</td><td>X</td><td></td><td>X</td><td>X</td></tr><tr><td>1</td><td>X</td><td>X</td><td></td><td></td><td></td><td></td><td>X</td><td>X</td><td>X</td><td></td><td></td><td>X</td><td>X</td><td>X</td><td>X</td><td></td><td>X</td></tr><tr><td>2</td><td>X</td><td>X</td><td>X</td><td></td><td></td><td></td><td></td><td>X</td><td>X</td><td>X</td><td></td><td></td><td>X</td><td></td><td>X</td><td>X</td><td>X</td></tr><tr><td>3</td><td></td><td>X</td><td>X</td><td>X</td><td></td><td></td><td></td><td>X</td><td>X</td><td>X</td><td>X</td><td></td><td></td><td>X</td><td></td><td>X</td><td>X</td></tr><tr><td>4</td><td></td><td></td><td>X</td><td>X</td><td>X</td><td></td><td></td><td></td><td>X</td><td>X</td><td>X</td><td>X</td><td></td><td></td><td>X</td><td>X</td><td>X</td></tr><tr><td>5</td><td></td><td></td><td></td><td>X</td><td>X</td><td>X</td><td></td><td></td><td></td><td>X</td><td>X</td><td>X</td><td>X</td><td></td><td></td><td>X</td><td>X</td></tr></table> <p>TABLE 1 EACH COLUMN INDICATES WHICH FEATURE MAP IN S2 ARE COMBINED BY THE UNITS IN A PARTICULAR FEATURE MAP OF C3.</p>		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0	X					X	X	X			X	X	X	X		X	X	1	X	X					X	X	X			X	X	X	X		X	2	X	X	X					X	X	X			X		X	X	X	3		X	X	X				X	X	X	X			X		X	X	4			X	X	X				X	X	X	X			X	X	X	5				X	X	X				X	X	X	X			X	X			
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15																																																																																																																
0	X					X	X	X			X	X	X	X		X	X																																																																																																															
1	X	X					X	X	X			X	X	X	X		X																																																																																																															
2	X	X	X					X	X	X			X		X	X	X																																																																																																															
3		X	X	X				X	X	X	X			X		X	X																																																																																																															
4			X	X	X				X	X	X	X			X	X	X																																																																																																															
5				X	X	X				X	X	X	X			X	X																																																																																																															
C3 레이어에서 6개의 입력을 16개의 피쳐 맵으로 변환하여 연산량을 줄이고, 글로벌 피쳐를 효율적으로 만들어냄.	VGG-16은 모든 컨볼루션 레이어에서 3x3 필터를 사용하고, 층을 깊게 쌓아 높은 표현력을 유지.	Depth-wise와 Point-wise 컨볼루션을 사용하여 매개변수와 연산 요구량을 크게 줄임.	Grouped Convolution과 채널 셔플링을 통해 연산 효율성을 높임.																																																																																																																													
모델이 단순하고 연산량이 적어 EMNIST 데이터셋에 적합하며, 효율적으로 중요한 피쳐를 추출할 수 있음.	EMNIST 데이터셋에서는 다각도 추출보다는 높은 정확도가 중요하므로, 깊은 네트워크 구조를 통해 높은 정확도를 달성할 수 있음.	EMNIST 데이터셋에 많은 매개변수를 가진 모델을 적용하는 것이 비효율적이므로, 경량화된 모델이 적합함.	EMNIST 이미지에서 글씨의 선형성을 고려하여, 그룹화된 컨볼루션을 통해 효율적으로 학습할 수 있음.																																																																																																																													



## Select the Model (모델 후보)

- EMNIST 데이터셋의 특성, 모델 구조 및 특성을 함께 고려하여 모델 후보 4개 선정

LeNet-5	VGG-16	MobileNet	ShuffleNet
 <p>TABLE 1 EACH COLUMN INDICATES WHICH FEATURE MAP IN S2 ARE COMBINED BY THE UNITS IN A PARTICULAR FEATURE MAP OF C3.</p>	 <p>convolution+ReLU max pooling fully nected+ReLU softmax</p>		
C3 레이어에서 6개의 입력을 16개의 피쳐 맵으로 변환하여 연산량을 줄이고, 글로벌 피쳐를 효율적으로 만들어냄.	VGG-16은 모든 컨볼루션 레이어에서 3x3 필터를 사용하고, 층을 깊게 쌓아 높은 표현력을 유지.	Depth-wise와 Point-wise 컨볼루션을 사용하여 매개변수와 연산 요구량을 크게 줄임.	Grouped Convolution과 채널 셔플링을 통해 연산 효율성을 높임.
모델이 단순하고 연산량이 적어 EMNIST 데이터셋에 적합하며, 효율적으로 중요한 피쳐를 추출할 수 있음.	EMNIST 데이터셋에서는 다각도 추출보다는 높은 정확도가 중요하므로, 깊은 네트워크 구조를 통해 높은 정확도를 달성할 수 있음.	EMNIST 데이터셋에 많은 매개변수를 가진 모델을 적용하는 것이 비효율적이므로, 경량화된 모델이 적합함.	EMNIST 이미지에서 글씨의 선형성을 고려하여, 그룹화된 컨볼루션을 통해 효율적으로 학습할 수 있음.

## Select the Model (추가 모델 후보 선정)

- 교재 + tensorflow 제공 모델 +  $\alpha$  (출시 연도 기준 정렬)

모델	출시 연도	크기 (MB)	파라미터 수	Top-1 정확도	Top-5 정확도	GPU(ms)	주요 특징
LeNet-5	1998	-	60K	-	-	-	초기 CNN 모델, 손글씨 숫자 인식
AlexNet	2012	240	60M	62.50%	83.00%	-	ImageNet 우승, ReLU, 드롭아웃, LRN 도입
GoogLeNet	2014	27	6.8M	74.80%	92.20%	-	인셉션 모듈, 계산 효율성 향상
VGG16	2014	528	138M	71.50%	89.80%	-	단순하고 깊은 구조, 3x3 컨볼루션 사용
Inception v3	2015	92	23.9M	77.90%	93.70%	6.9	병렬 컨볼루션, 차원 축소
ResNet50	2015	98	25.6M	76.00%	93.00%	4.4	잔차 연결, 깊은 네트워크
Inception-ResNet v2	2016	215	55.9M	80.30%	95.30%	10	Inception + ResNet 결합
SqueezeNet	2016	1.3	1.2M	57.50%	80.30%	-	모델 크기 축소, 'Fire' 모듈 사용
DarkNet	2016	-	23M	-	-	-	YOLO 백본 모델, 다양한 크기의 필터 사용
DenseNet121	2017	33	8M	74.90%	92.20%	-	밀집 연결, 특성 재사용
NASNetLarge	2017	343	88.9M	82.70%	96.20%	-	NAS 알고리즘 설계, 고성능
NASNetMobile	2017	23	5.3M	74.40%	91.90%	6.7	NAS 알고리즘 설계, 고성능
MobileNet	2017	16	4.3M	70.40%	89.50%	3.4	경량화된 아키텍처, 깊이별 분리 컨볼루션 사용
Xception	2017	88	22.9M	79.00%	94.50%	-	깊이별 분리 컨볼루션 확장, 효율적인 성능
SENet	2017	115	115M	82.70%	95.40%	-	Squeeze-and-Excitation 블록, 채널별 중요도 재조정
ShuffleNet	2017	5	1.3M	70.90%	-	-	경량화 모델, 채널 셔플 사용
MobileNetV2	2018	14	3.5M	71.30%	90.10%	3.8	경량화된 아키텍처, 깊이별 분리 컨볼루션 사용
EfficientNetB0	2019	29	5.3M	77.10%	93.30%	4.9	균형 잡힌 네트워크 스케일링
EfficientNetB1	2019	31	7.9M	79.10%	94.40%	5.6	균형 잡힌 네트워크 스케일링
EfficientNetB2	2019	36	9.2M	80.10%	94.90%	6.5	균형 잡힌 네트워크 스케일링
EfficientNetB3	2019	48	12.3M	81.60%	95.70%	8.8	균형 잡힌 네트워크 스케일링
EfficientNetB4	2019	75	19.5M	82.90%	96.40%	15.1	균형 잡힌 네트워크 스케일링
EfficientNetB5	2019	118	30.6M	83.60%	96.70%	25.3	균형 잡힌 네트워크 스케일링
EfficientNetB6	2019	166	43.3M	84.00%	96.80%	40.4	균형 잡힌 네트워크 스케일링
EfficientNetB7	2019	256	66.7M	84.30%	97.00%	61.6	균형 잡힌 네트워크 스케일링
CSPNet	2019	-	-	-	-	-	스테이지 간 특징 맵 부분 연결, 학습 효율성 최적화
RegNet	2020	-	11.2M	77.00%	-	-	규칙적인 네트워크 구조, 다양한 모델 크기
ConvNeXtBase	2021	328	88M	82.90%	-	-	Transformer 기법에서 영감, 다양한 크기로 구현 가능

## Select the Model (추가 모델 후보 선정)

- 교재 + tensorflow 제공 모델 +  $\alpha$  (파라미터 수 기준 정렬)

모델	출시 연도	크기 (MB)	파라미터 수	Top-1 정확도	Top-5 정확도	GPU(ms)	주요 특징
LeNet-5	1998	-	60K	-	-	-	초기 CNN 모델, 손글씨 숫자 인식
SqueezeNet	2016	1.3	1.2M	57.50%	80.30%	-	모델 크기 축소, 'Fire' 모듈 사용
ShuffleNet	2017	5	1.3M	70.90%	-	-	경량화 모델, 채널 셔플 사용
MobileNetV2	2018	14	3.5M	71.30%	90.10%	3.8	경량화된 아키텍처, 깊이별 분리 컨볼루션 사용
MobileNet	2017	16	4.3M	70.40%	89.50%	3.4	경량화된 아키텍처, 깊이별 분리 컨볼루션 사용
NASNetMobile	2017	23	5.3M	74.40%	91.90%	6.7	NAS 알고리즘 설계, 고성능
EfficientNetB0	2019	29	5.3M	77.10%	93.30%	4.9	균형 잡힌 네트워크 스케일링
GoogLeNet	2014	27	6.8M	74.80%	92.20%	-	인셉션 모듈, 계산 효율성 향상
EfficientNetB1	2019	31	7.9M	79.10%	94.40%	5.6	균형 잡힌 네트워크 스케일링
DenseNet121	2017	33	8M	74.90%	92.20%	-	밀집 연결, 특성 재사용
EfficientNetB2	2019	36	9.2M	80.10%	94.90%	6.5	균형 잡힌 네트워크 스케일링
RegNet	2020	-	11.2M	77.00%	-	-	규칙적인 네트워크 구조, 다양한 모델 크기
EfficientNetB3	2019	48	12.3M	81.60%	95.70%	8.8	균형 잡힌 네트워크 스케일링
Inception v3	2015	92	23.9M	77.90%	93.70%	6.9	병렬 컨볼루션, 차원 축소
DarkNet	2016	-	23M	-	-	-	YOLO 백본 모델, 다양한 크기의 필터 사용
ResNet50	2015	98	25.6M	76.00%	93.00%	4.4	잔차 연결, 깊은 네트워크
Xception	2017	88	22.9M	79.00%	94.50%	-	깊이별 분리 컨볼루션 확장, 효율적인 성능
EfficientNetB4	2019	75	19.5M	82.90%	96.40%	15.1	균형 잡힌 네트워크 스케일링
EfficientNetB5	2019	118	30.6M	83.60%	96.70%	25.3	균형 잡힌 네트워크 스케일링
EfficientNetB6	2019	166	43.3M	84.00%	96.80%	40.4	균형 잡힌 네트워크 스케일링
Inception-ResNet v2	2016	215	55.9M	80.30%	95.30%	10	Inception + ResNet 결합
EfficientNetB7	2019	256	66.7M	84.30%	97.00%	61.6	균형 잡힌 네트워크 스케일링
AlexNet	2012	240	60M	62.50%	83.00%	-	ImageNet 우승, ReLU, 드롭아웃, LRN 도입
ConvNeXtBase	2021	328	88M	82.90%	-	-	Transformer 기법에서 영감, 다양한 크기로 구현 가능
NASNetLarge	2017	343	88.9M	82.70%	96.20%	-	NAS 알고리즘 설계, 고성능
SENet	2017	115	115M	82.70%	95.40%	-	Squeeze-and-Excitation 블록, 채널별 중요도 재조정
VGG16	2014	528	138M	71.50%	89.80%	-	단순하고 깊은 구조, 3x3 컨볼루션 사용
CSPNet	2019	-	-	-	-	-	스테이지 간 특징 맵 부분 연결, 학습 효율성 최적화

## Select the Model (추가 모델 후보 선정)

- 교재 + tensorflow 제공 모델 +  $\alpha$  (GPU 속도 기준 정렬)

모델	출시 연도	크기 (MB)	파라미터 수	Top-1 정확도	Top-5 정확도	GPU(ms)	주요 특징
MobileNet	2017	16	4.3M	70.40%	89.50%	3.4	경량화된 아키텍처, 깊이별 분리 컨볼루션 사용
MobileNetV2	2018	14	3.5M	71.30%	90.10%	3.8	경량화된 아키텍처, 깊이별 분리 컨볼루션 사용
ResNet50	2015	98	25.6M	76.00%	93.00%	4.4	잔차 연결, 깊은 네트워크
EfficientNetB0	2019	29	5.3M	77.10%	93.30%	4.9	균형 잡힌 네트워크 스케일링
EfficientNetB1	2019	31	7.9M	79.10%	94.40%	5.6	균형 잡힌 네트워크 스케일링
EfficientNetB2	2019	36	9.2M	80.10%	94.90%	6.5	균형 잡힌 네트워크 스케일링
NASNetMobile	2017	23	5.3M	74.40%	91.90%	6.7	NAS 알고리즘 설계, 고성능
Inception v3	2015	92	23.9M	77.90%	93.70%	6.9	병렬 컨볼루션, 차원 축소
EfficientNetB3	2019	48	12.3M	81.60%	95.70%	8.8	균형 잡힌 네트워크 스케일링
Inception-ResNet v2	2016	215	55.9M	80.30%	95.30%	10	Inception + ResNet 결합
EfficientNetB4	2019	75	19.5M	82.90%	96.40%	15.1	균형 잡힌 네트워크 스케일링
EfficientNetB5	2019	118	30.6M	83.60%	96.70%	25.3	균형 잡힌 네트워크 스케일링
EfficientNetB6	2019	166	43.3M	84.00%	96.80%	40.4	균형 잡힌 네트워크 스케일링
EfficientNetB7	2019	256	66.7M	84.30%	97.00%	61.6	균형 잡힌 네트워크 스케일링
LeNet-5	1998	-	60K	-	-	-	초기 CNN 모델, 손글씨 숫자 인식
SqueezeNet	2016	1.3	1.2M	57.50%	80.30%	-	모델 크기 축소, 'Fire' 모듈 사용
ShuffleNet	2017	5	1.3M	70.90%	-	-	경량화 모델, 채널 셔플 사용
DenseNet121	2017	33	8M	74.90%	92.20%	-	밀집 연결, 특성 재사용
GoogLeNet	2014	27	6.8M	74.80%	92.20%	-	인셉션 모듈, 계산 효율성 향상
NASNetLarge	2017	343	88.9M	82.70%	96.20%	-	NAS 알고리즘 설계, 고성능
SENet	2017	115	115M	82.70%	95.40%	-	Squeeze-and-Excitation 블록, 채널별 중요도 재조정
Xception	2017	88	22.9M	79.00%	94.50%	-	깊이별 분리 컨볼루션 확장, 효율적인 성능
RegNet	2020	-	11.2M	77.00%	-	-	규칙적인 네트워크 구조, 다양한 모델 크기
AlexNet	2012	240	60M	62.50%	83.00%	-	ImageNet 우승, ReLU, 드롭아웃, LRN 도입
DarkNet	2016	-	23M	-	-	-	YOLO 백본 모델, 다양한 크기의 필터 사용
CSPNet	2019	-	-	-	-	-	스테이지 간 특징 맵 부분 연결, 학습 효율성 최적화
ConvNeXtBase	2021	328	88M	82.90%	-	-	Transformer 기법에서 영감, 다양한 크기로 구현 가능
VGG16	2014	528	138M	71.50%	89.80%	-	단순하고 깊은 구조, 3x3 컨볼루션 사용

## Select the Model (추가 모델 후보 선정)

- 교재 + tensorflow 제공 모델 +  $\alpha$  (Top-5 정확도 기준 정렬)

모델	출시 연도	크기 (MB)	파라미터 수	Top-1 정확도	Top-5 정확도	GPU(ms)	주요 특징
EfficientNetB7	2019	256	66.7M	84.30%	97.00%	61.6	균형 잡힌 네트워크 스케일링
EfficientNetB6	2019	166	43.3M	84.00%	96.80%	40.4	균형 잡힌 네트워크 스케일링
EfficientNetB5	2019	118	30.6M	83.60%	96.70%	25.3	균형 잡힌 네트워크 스케일링
EfficientNetB4	2019	75	19.5M	82.90%	96.40%	15.1	균형 잡힌 네트워크 스케일링
NASNetLarge	2017	343	88.9M	82.70%	96.20%	-	NAS 알고리즘 설계, 고성능
SENet	2017	115	115M	82.70%	95.40%	-	Squeeze-and-Excitation 블록, 채널별 중요도 재조정
EfficientNetB3	2019	48	12.3M	81.60%	95.70%	8.8	균형 잡힌 네트워크 스케일링
Inception-ResNet v2	2016	215	55.9M	80.30%	95.30%	10	Inception + ResNet 결합
EfficientNetB2	2019	36	9.2M	80.10%	94.90%	6.5	균형 잡힌 네트워크 스케일링
Xception	2017	88	22.9M	79.00%	94.50%	-	깊이별 분리 컨볼루션 확장, 효율적인 성능
EfficientNetB1	2019	31	7.9M	79.10%	94.40%	5.6	균형 잡힌 네트워크 스케일링
Inception v3	2015	92	23.9M	77.90%	93.70%	6.9	병렬 컨볼루션, 차원 축소
EfficientNetB0	2019	29	5.3M	77.10%	93.30%	4.9	균형 잡힌 네트워크 스케일링
ResNet50	2015	98	25.6M	76.00%	93.00%	4.4	잔차 연결, 깊은 네트워크
GoogLeNet	2014	27	6.8M	74.80%	92.20%	-	인셉션 모듈, 계산 효율성 향상
DenseNet121	2017	33	8M	74.90%	92.20%	-	밀집 연결, 특성 재사용
NASNetMobile	2017	23	5.3M	74.40%	91.90%	6.7	NAS 알고리즘 설계, 고성능
MobileNetV2	2018	14	3.5M	71.30%	90.10%	3.8	경량화된 아키텍처, 깊이별 분리 컨볼루션 사용
MobileNet	2017	16	4.3M	70.40%	89.50%	3.4	경량화된 아키텍처, 깊이별 분리 컨볼루션 사용
VGG16	2014	528	138M	71.50%	89.80%	-	단순하고 깊은 구조, 3x3 컨볼루션 사용
ShuffleNet	2017	5	1.3M	70.90%	-	-	경량화 모델, 채널 셔플 사용
AlexNet	2012	240	60M	62.50%	83.00%	-	ImageNet 우승, ReLU, 드롭아웃, LRN 도입
SqueezeNet	2016	1.3	1.2M	57.50%	80.30%	-	모델 크기 축소, 'Fire' 모듈 사용
LeNet-5	1998	-	60K	-	-	-	초기 CNN 모델, 손글씨 숫자 인식
DarkNet	2016	-	23M	-	-	-	YOLO 백본 모델, 다양한 크기의 필터 사용
RegNet	2020	-	11.2M	77.00%	-	-	규칙적인 네트워크 구조, 다양한 모델 크기
CSPNet	2019	-	-	-	-	-	스테이지 간 특징 맵 부분 연결, 학습 효율성 최적화
ConvNeXtBase	2021	328	88M	82.90%	-	-	Transformer 기법에서 영감, 다양한 크기로 구현 가능

## Select the Model (추가 모델 후보 선정)

모델	top-5 정확도	정규화	모델	GPU(ms)	정규화	모델	파라미터 수	정규화된 값	모델	합산 점수
EfficientNetB7	97.00%	1	MobileNet	3.4	1	LeNet-5	60K	1	EfficientNetB1	2.74
EfficientNetB6	96.80%	0.99	MobileNetV2	3.8	0.99	SqueezeNet	1.2M	0.99	EfficientNetB2	2.74
EfficientNetB5	96.70%	0.98	ResNet50	4.4	0.98	ShuffleNet	1.3M	0.99	EfficientNetB0	2.71
EfficientNetB4	96.40%	0.96	EfficientNetB0	4.9	0.97	MobileNetV2	3.5M	0.97	EfficientNetB3	2.71
NASNetLarge	96.20%	0.95	EfficientNetB1	5.6	0.96	MobileNet	4.3M	0.97	Inception v3	2.58
SENet	95.40%	0.92	EfficientNetB2	6.5	0.94	NASNetMobile	5.3M	0.96	MobileNetV2	2.55
EfficientNetB3	95.70%	0.9	NASNetMobile	6.7	0.94	EfficientNetB0	5.3M	0.96	ResNet50	2.56
Inception-ResNet v2	95.30%	0.9	Inception v3	6.9	0.94	GoogLeNet	6.8M	0.95	MobileNet	2.52
EfficientNetB2	94.90%	0.87	EfficientNetB3	8.8	0.9	EfficientNetB1	7.9M	0.94	NASNetMobile	2.59
Xception	94.50%	0.85	Inception-ResNet v2	10	0.88	DenseNet121	8M	0.94	EfficientNetB4	2.62
EfficientNetB1	94.40%	0.84	EfficientNetB4	15.1	0.8	EfficientNetB2	9.2M	0.93	EfficientNetB5	2.37
Inception v3	93.70%	0.8	EfficientNetB5	25.3	0.61	RegNet	11.2M	0.92	Inception-ResNet v2	2.38
EfficientNetB0	93.30%	0.78	EfficientNetB6	40.4	0.34	EfficientNetB3	12.3M	0.91	EfficientNetB6	2.02
ResNet50	93.00%	0.76	EfficientNetB7	61.6	0	Inception v3	23.9M	0.84	EfficientNetB7	1.52
GoogLeNet	92.20%	0.71	LeNet-5	-	0	DarkNet	23M	0.84		
DenseNet121	92.20%	0.71	SqueezeNet	-	0	ResNet50	25.6M	0.82		
NASNetMobile	91.90%	0.69	ShuffleNet	-	0	Xception	22.9M	0.84		
MobileNetV2	90.10%	0.59	DenseNet121	-	0	EfficientNetB4	19.5M	0.86		
MobileNet	89.50%	0.55	GoogLeNet	-	0	EfficientNetB5	30.6M	0.78		
VGG16	89.80%	0.57	NASNetLarge	-	0	EfficientNetB6	43.3M	0.69		
ShuffleNet	-		SENet	-	0	Inception-ResNet v2	55.9M	0.6		
AlexNet	83.00%	0.16	Xception	-	0	AlexNet	60M	0.57		
SqueezeNet	80.30%	0	RegNet	-	0	EfficientNetB7	66.7M	0.52		
LeNet-5	-		AlexNet	-	0	ConvNeXtBase	88M	0.36		
DarkNet	-		DarkNet	-	0	NASNetLarge	88.9M	0.35		
RegNet	-		CSPNet	-	0	SENet	115M	0.18		
CSPNet	-		ConvNeXtBase	-	0	VGG16	138M	0		
ConvNeXtBase	-		VGG16	-	0	CSPNet	-	0		



# Plan

1. 최종 모델 선정 → 통계적 가설검정

2. 하이퍼파라미터

1. 배치 사이즈 변경

2. Learning Rate Scheduler

3. 활성화 함수 변경 → 더 다양한 함수 활용

4. Regularization 추가 → 더 다양한 규제 방법 활용

5. 데이터 증강(data augmentation) → 더 다양한 augmentation 방법 활용

6. optimizer 변경

7. batchnormalization 추가

8. 네트워크 깊이 및 폭 변경: 블록 수 변경, 각 블록의 필터 수 변경

감사합니다