CNN-Based Model for Recognizing Street View House Numbers(SVHN)

- Shiwei Yu, Jiaying Zhuang, Xiaomeng Chen, Zixuan Wang, Zidong Xu -

CONTENTS

01

Introduction & Dataset

Introduction

Our purposes:

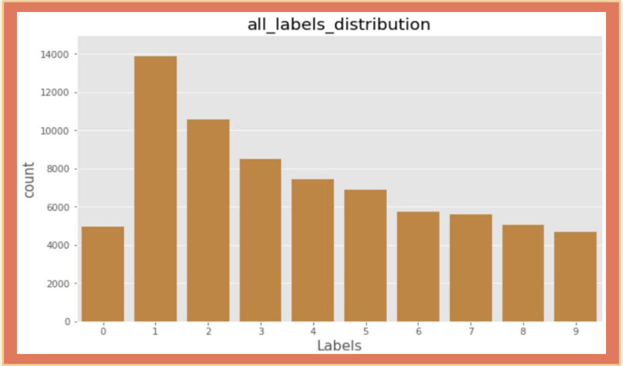
- Model development - address a subset of digit recognition using the Google Street View House Numbers dataset.

- Practical Applications - update map info automatically.

House Numbers (SVHN) Dataset

Dataset

- **Overview**
  10 classes, 1 for each digit. Digit '1' has label 1, '9' has label 9 and '0' has label 10
  73257 digits for training



- The **format** we use: MNIST-like 32-by-32 images centered around a single character

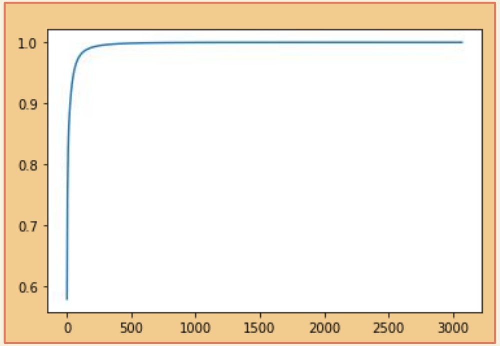
02

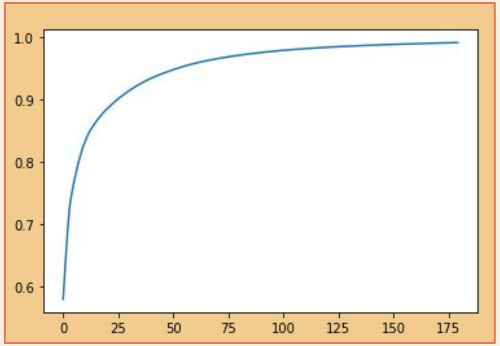
Models & Results

PCA & KNN

Our expectation:

- It can reduce the problems caused by the disaster of dimensionality (high dimension);
- It can be used to compress data and minimize loss data;
- It can reduce high-dimensional data to low-dimensional for visualization.





After PCA dimensionality reduction, the first 173 features can represent 99% of the original features

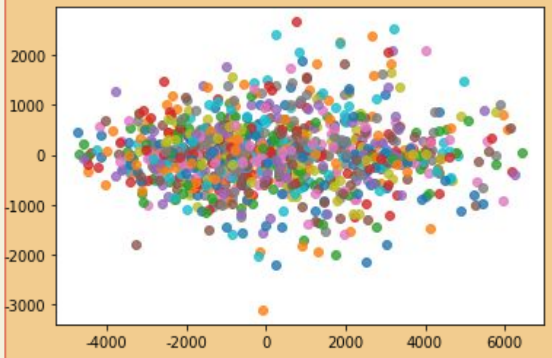
PCA & KNN

The PCA results are clustered and presented by KNN:

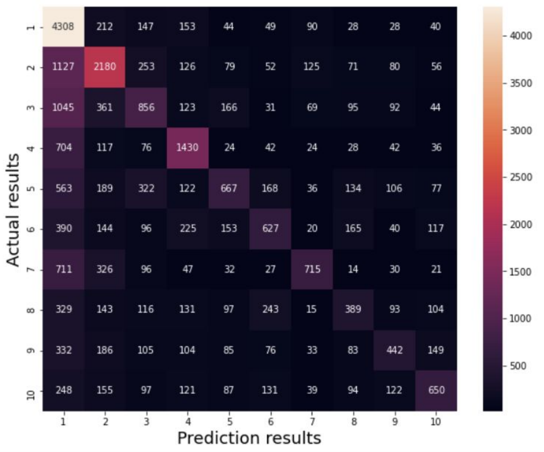
```
y_true=y_test.tolist()
y_pred=pre_label.tolist()
print(classification_report(y_true,y_pred))
print('Accuracy on the test set: %.2f'% accuracy_score(y_true,y_pred))
```

	precision	recall	f1-score	support
1	0.44	0.84	0.58	5099
2	0.54	0.53	0.53	4149
3	0.40	0.30	0.34	2882
4	0.55	0.57	0.56	2523
5	0.47	0.28	0.35	2384
6	0.43	0.32	0.37	1977
7	0.61	0.35	0.45	2019
8	0.35	0.23	0.28	1660
9	0.41	0.28	0.33	1595
10	0.50	0.37	0.43	1744
accuracy			0.47	26032
macro avg	0.47	0.41	0.42	26032
weighted avg	0.47	0.47	0.45	26032

Accuracy on the test set: 0.47

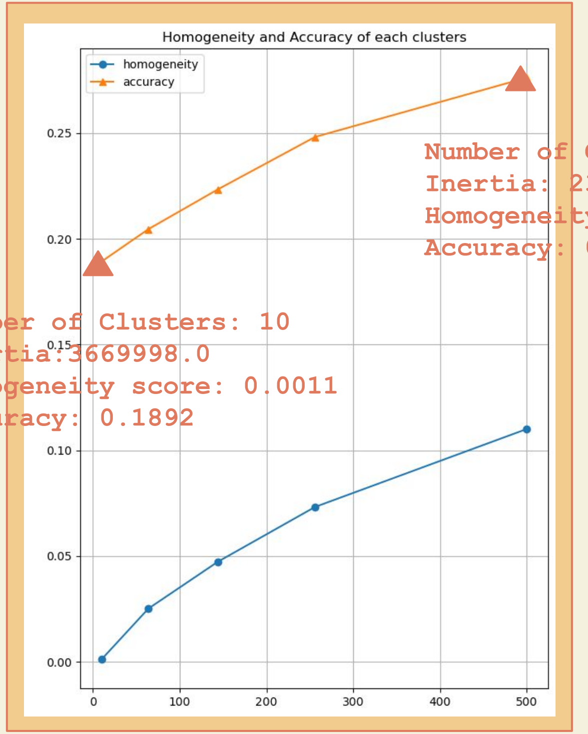



```
<AxesSubplot:>
Text(69.0, 0.5, 'Actual results')
Text(0.5, 51.0, 'Prediction results')
```



K-Means

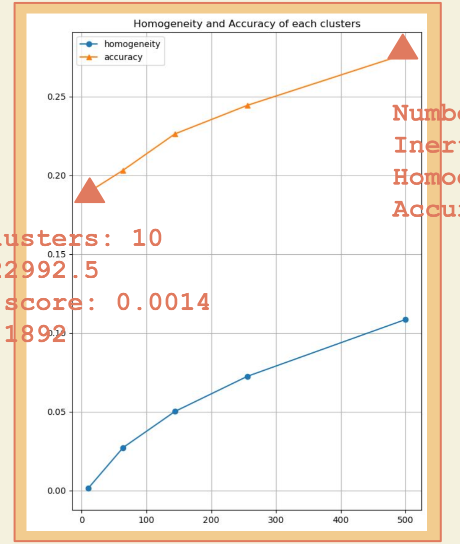
- Before reducing dimensionality



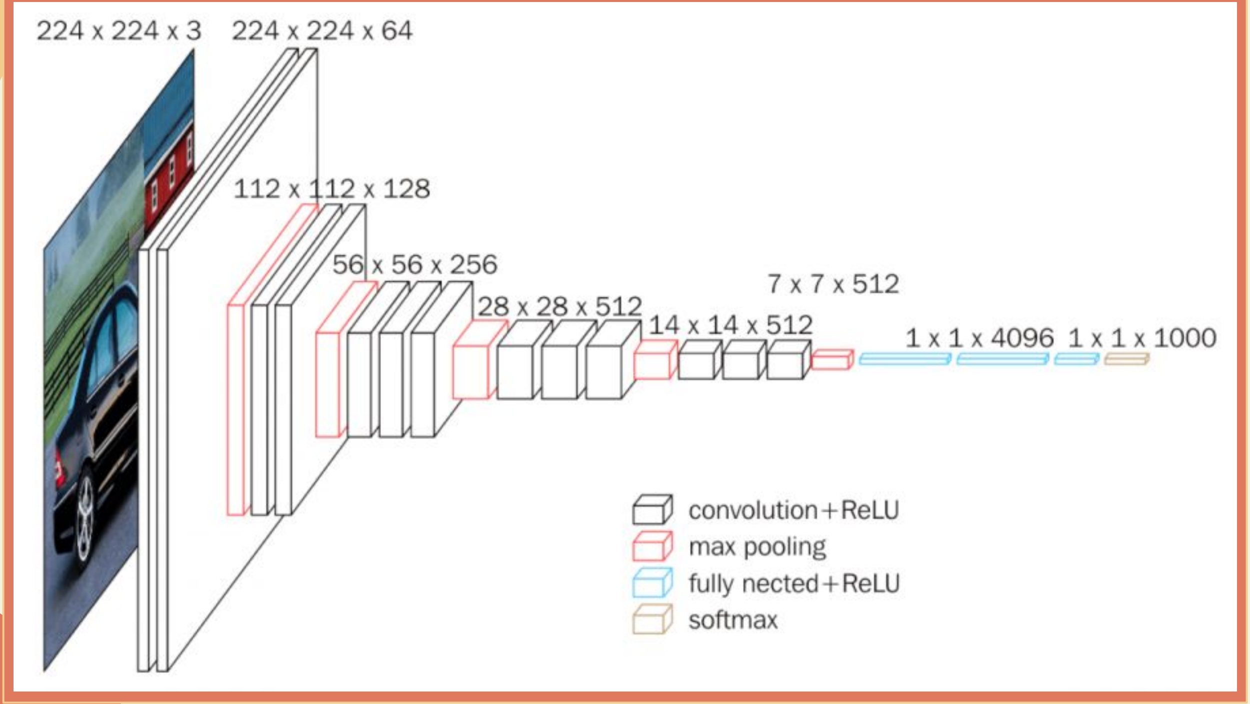
- After reducing dimensionality

We compress the data using **PCA** to a degree that preserves 95% variance of the data and only lost 5%

Only **54** out of 3072 features can preserve 95% of the data, which means SVHN is originally very sparse and most of the data is rather present at a much lower dimension



Pretrained VGG16

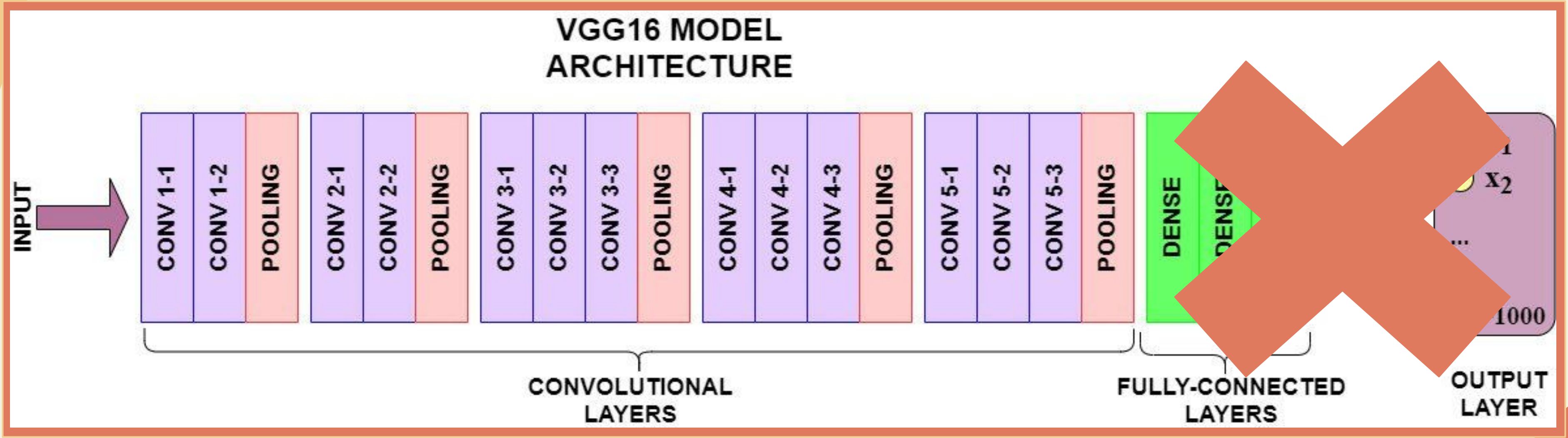


CNN
Large-scale Image Recognition
Pretrained on ImageNet

Architecture:

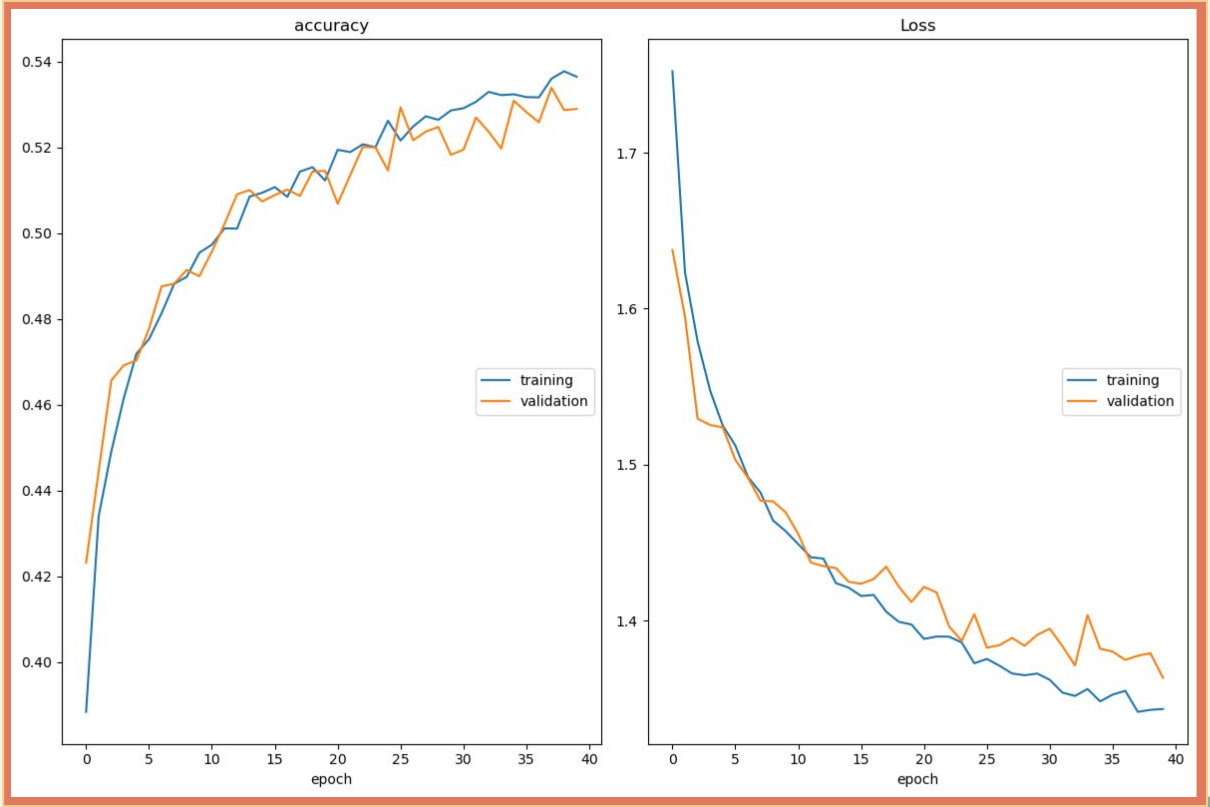
Input: 3-channel
13 convolutional layers
5 max-pooling layers
3 fully-connected layers
Activation: ReLU & softmax

VGG16 without Fine-Tuning



Remove the “top” portion of the model.
Load all the pretrained weights of ImageNet.
Customized fully-connected layers.
Output: 10 classes

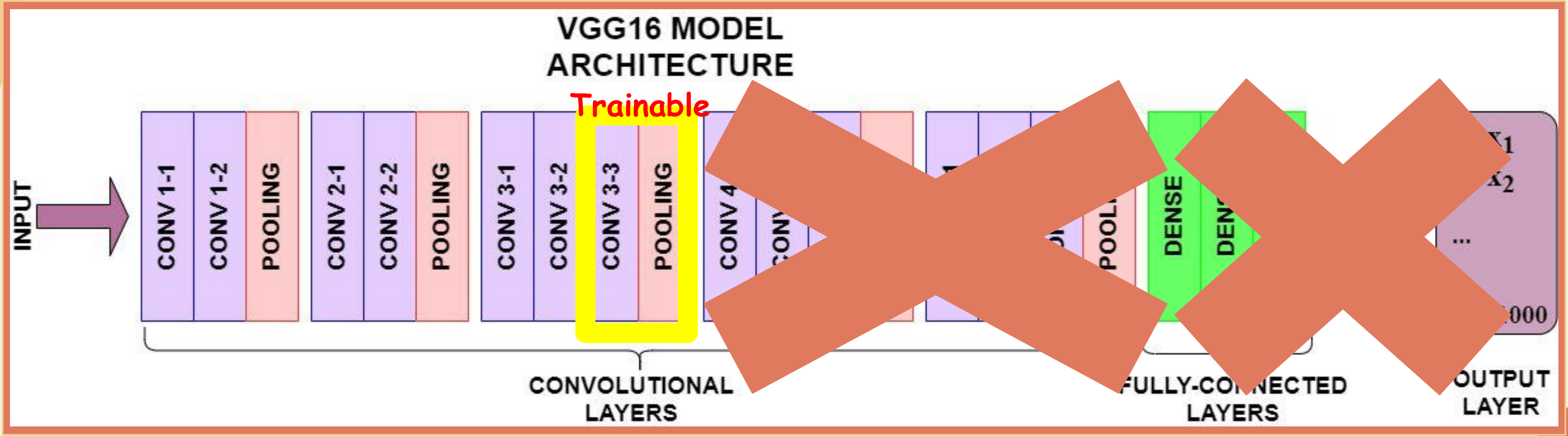
VGG16 without Fine-Tuning



Test Acc: 40.13%

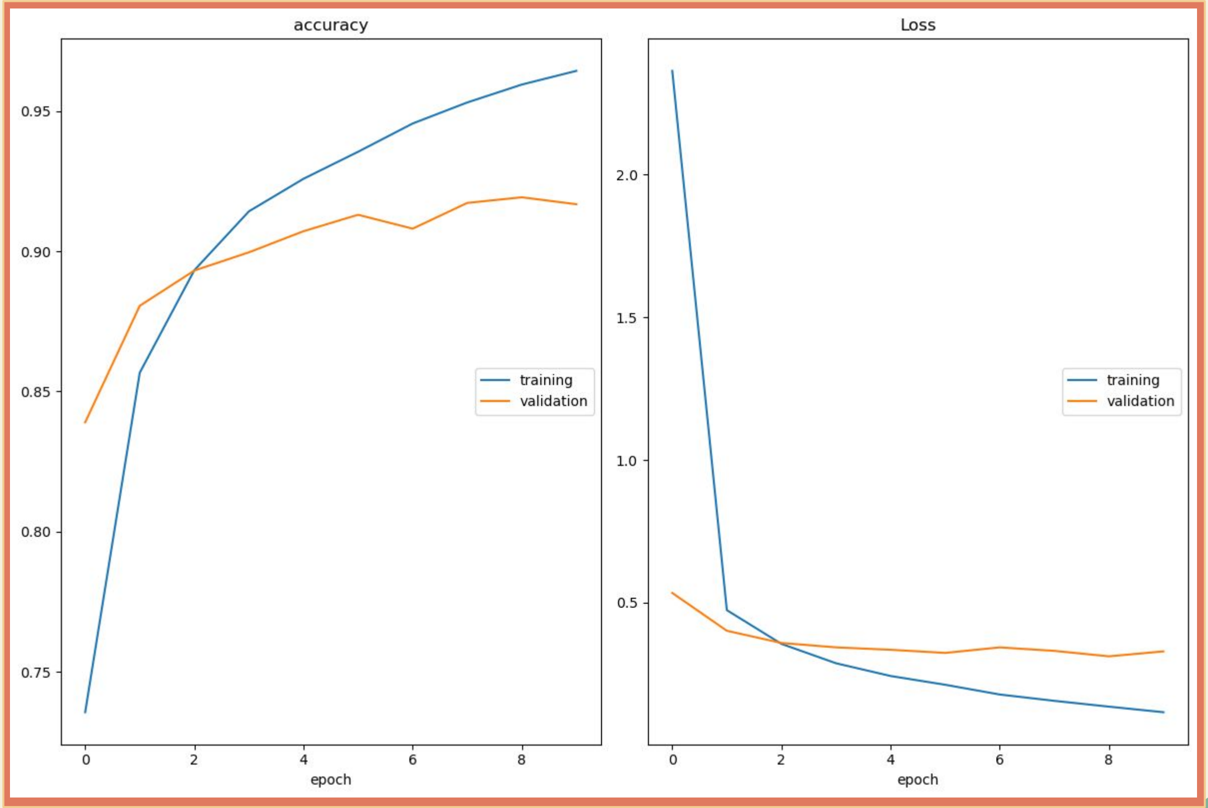
Retrain convolutional layers
Load less pretrained weights of layers

VGG16 with Fine-Tuning



Take only the first 10 layers.
Freeze the first 8 layers and retrain the last convolutional layer and pooling layer.

VGG16 with Fine-Tuning

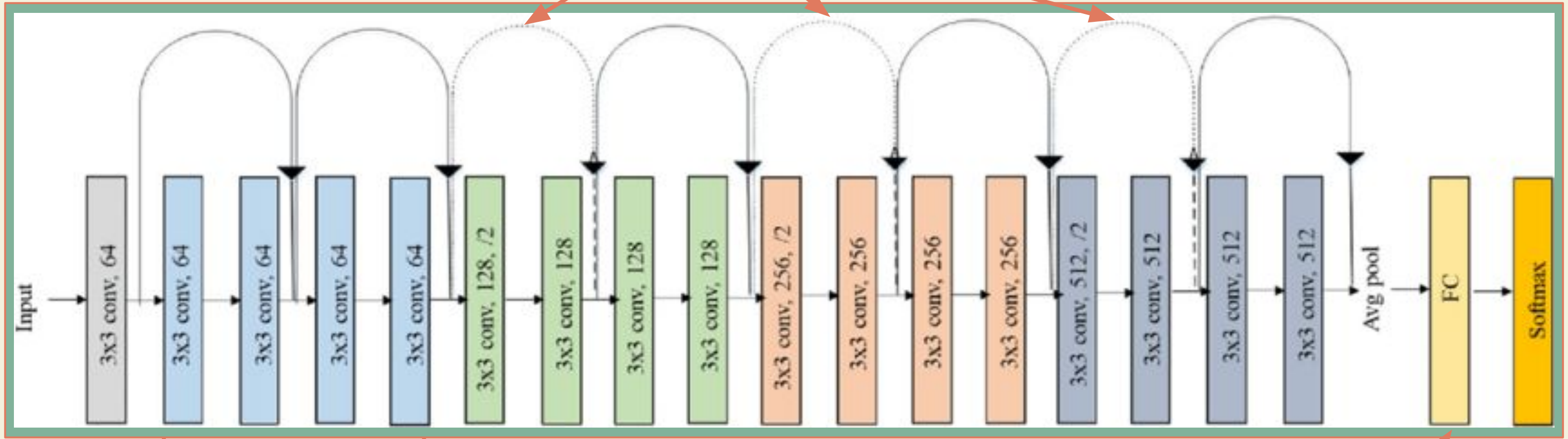


Test Acc: 92.42%

ResNet-18

ORIGINAL STRUCTURE

SKIP CONNECTION



BLOCK 1

BLOCK 2

BLOCK 3

BLOCK 4

FULLY CONNECTED

IDENTICAL ConvNets

ResNet

ResNet-18-Based STRUCTURE

Transform: torch.Size([3, 32, 32])

Torchvision.RandomCrop & RandomRotation

INPUT SHAPE (10,3,28,28)

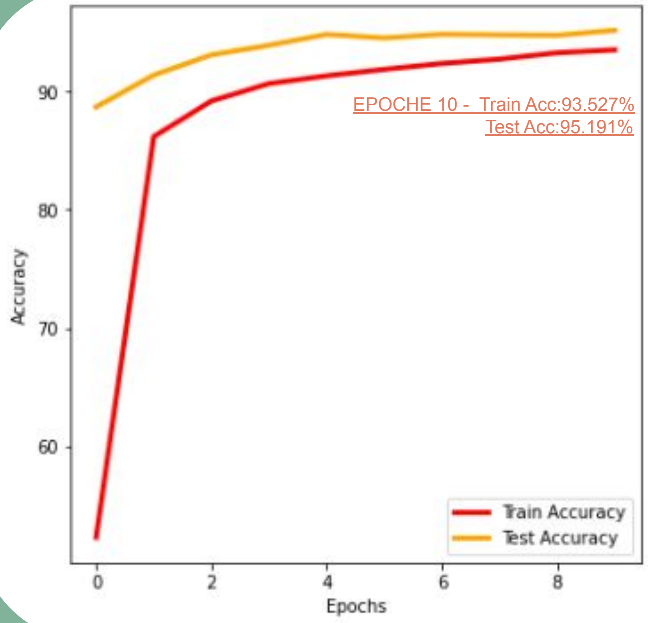
SELF BLOCK 1

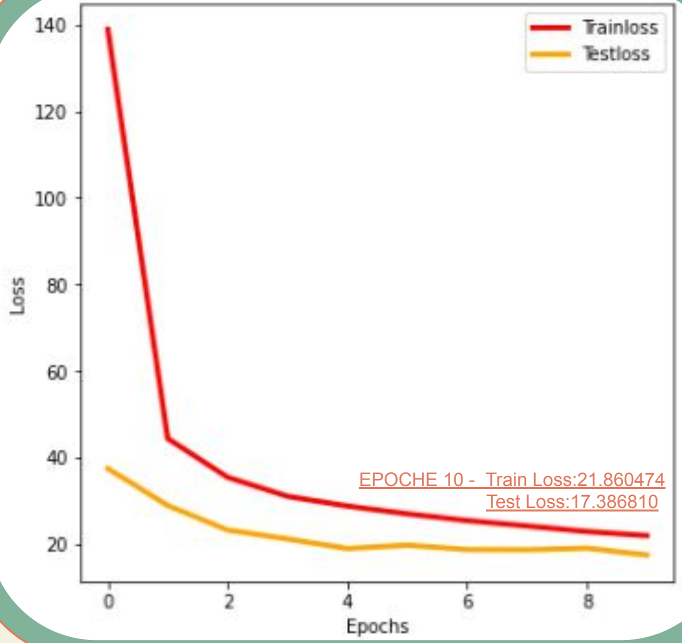
SELF BLOCK 2

SELF BLOCK 3

```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
MyResNet                                 [10, 10]                  --
├─Sequential: 1-1                        [10, 64, 28, 28]          --
│    └─Conv2d: 2-1                       [10, 64, 28, 28]          1,728
│    └─BatchNorm2d: 2-2                  [10, 64, 28, 28]          128
│    └─ReLU: 2-3                         [10, 64, 28, 28]          --
├─Sequential: 1-2                        [10, 128, 14, 14]         --
│    └─BasicBlock: 2-4                   [10, 128, 14, 14]         230,144
│    └─BasicBlock: 2-5                   [10, 128, 14, 14]         295,424
├─Sequential: 1-3                        [10, 256, 7, 7]           --
│    └─BasicBlock: 2-6                   [10, 256, 7, 7]           919,040
│    └─BasicBlock: 2-7                   [10, 256, 7, 7]           1,180,672
├─AdaptiveAvgPool2d: 1-4                 [10, 256, 1, 1]           --
├─Linear: 1-5                            [10, 10]                  2,570
==========================================================================================
Total params: 2,629,706
Trainable params: 2,629,706
Non-trainable params: 0
```

ResNet-18


EPOCHE 10 - Train Acc:93.527%
Test Acc:95.191%
Accuracy
Epochs
Train Accuracy
Test Accuracy
90
80
70
60
0
2
4
6
8


EPOCHE 10 - Train Loss:21.860474
Test Loss:17.386810
Loss
Epochs
Trainloss
Testloss
140
120
100
80
60
40
20
0
2
4
6
8

03

Conclusion

K-Means **27.7%**	• Dimensionality reduction can help only when cluster amounts are large • Both models are not performing well (27.61%, 27.7%)
PCA & KNN **49.0%**	• 49% accuracy • Parameter space of the image is sparse
VGG-16 **92.4%**	• Fine-tuning can make the model better fit the dataset • Utilizing only the first 10 layers performs better in digit detection in our case, probably because the full 16-layer version pretrained on ImageNet focuses more on object detection.
ResNet **95.2%**	• Good accuracy despite high loss • The losses on both the training and test sets show a decreasing trend → Increase the epochs until early stopping

Reference:

- https://neurohive.io/en/popular-networks/vgg16/
- https://www.learndatasci.com/tutorials/hands-on-transfer-learning-keras/
- https://towardsdatascience.com/transfer-learning-with-vgg16-and-keras-50ea161580b4
- https://blog.csdn.net/Grateful_Dead424/article/details/124292041
- https://www.researchgate.net/publication/336642248_A_Deep_Learning_Approach_for_Automated_Diagnosis_and_Multi-Class_Classification_of_Alzheimer's_Disease_Stages_Using_Resting-State_fMRI_and_Residual_Neural_Networks

Thanks for Watching