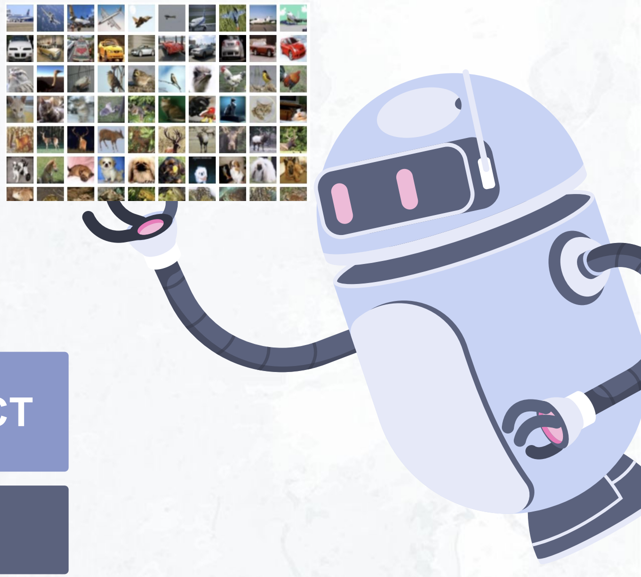
Image Classification on Tiny ImageNet



CS 172B NEURAL NETWORKS AND DEEP LEARNING PROJECT

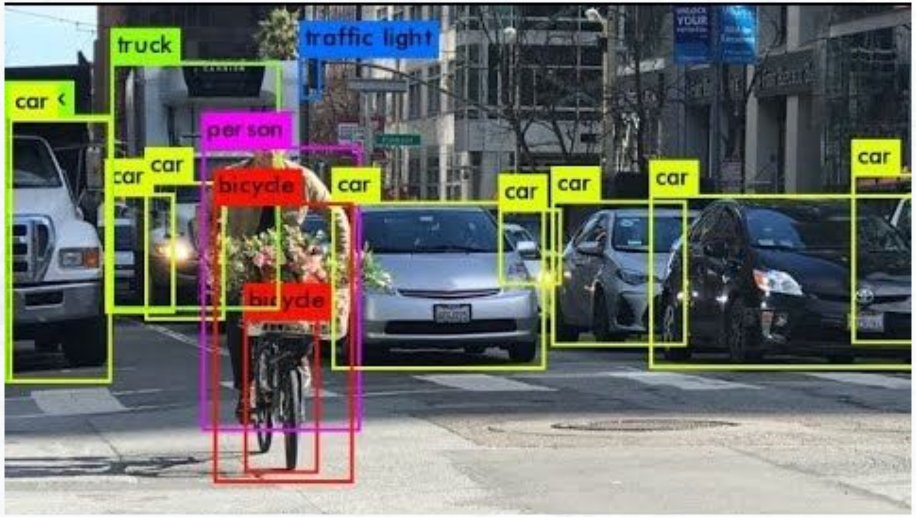
FRANK DONG, NICHOLAS PAVINI, TIANYUN YUAN, JINGQI YAO, NIMA HENDI

Table of contents

1

INTRODUCTION

The ImageNet Large Scale Visual Recognition Challenge(ILSVRC) started in 2010 and has become the standard benchmark of image recognition. Tiny ImageNet Challenge is a similar challenge with a smaller dataset and less image classes.



Previous Work

- AlexNet (2012)
  - 60 MILLION PARAMETERS AND 65,000 NEURONS, 5 CONVOLUTIONAL LAYERS
  - TOP-5 ERROR RATE OF 18.9% IN THE ILSVRC-2010 CONTEST
  - USES DATA AUGMENTATION AND DROPOUT TO FIGHT OVERFITTING
- VGGNet (2014)
  - DEEPER CNN WITH SMALLER FILTERS
  - TOP-5 TEST ERROR OF 6.8% IN ILSVRC-2014 CONTEST
  - HAVE DEGRADATION PROBLEM
- ResNet (2015)
  - PREVENT DEGRADATION OF DEEPER CNNS

Project Overview

- Distinguish performance difference between different image classification systems using regular neural networks, classic CNNs, Multi-Layer CNN, VGGNet16 and ResNet
- Apply data augmentation, dropout to prevent overfitting of our models
- Use transfer learning, layer-wise fine-tuning to initialize weight
- Present data visualization of performance between different models

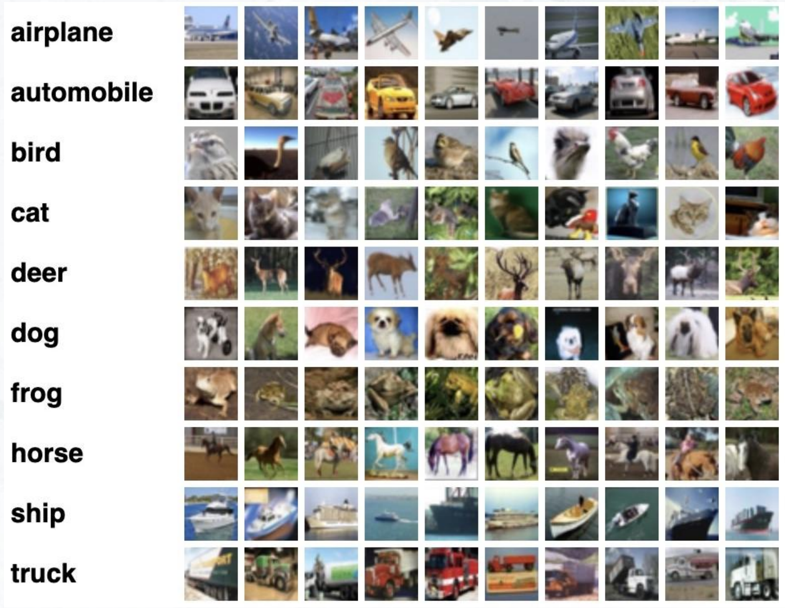
2

DATASET

What is Tiny ImageNet?

[define] Tiny ImageNet is a dataset containing 100,000 images of 200 classes (500 for each class) downsized to 64×64 colored images. Each class has 500 training images, 50 validation images and 50 test images. It serves as a benchmark for evaluating image classification models.

[use] Bigger and more challenging datasets can be modeled using Tiny ImageNet since it is easier to model.



Data Processing

PHASE 1

Scrap and wrangle the data from Tiny Imagenet

PHASE 2

Preprocess the data using Pandas and store them in the database

PHASE 3

Build machine learning models to classify pictures

Tiny ImageNet Dataset

Data PreProcess

We applied image transformation:

- Convert Images to tensors to process inside the network
- Create the dataset and the DataLoaders to benefit from batching and shuffling
- Change the directory tree to meeting the requirements of the data loader

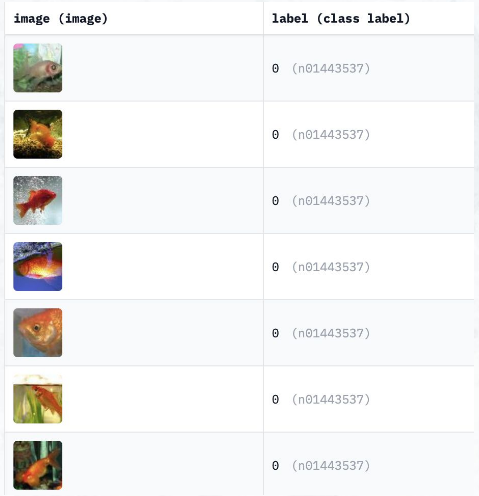
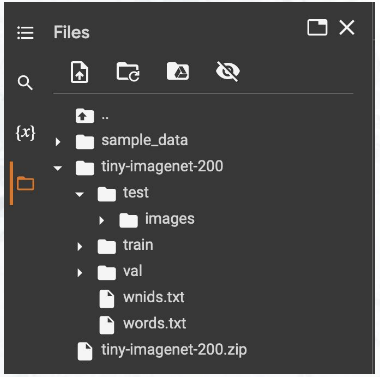
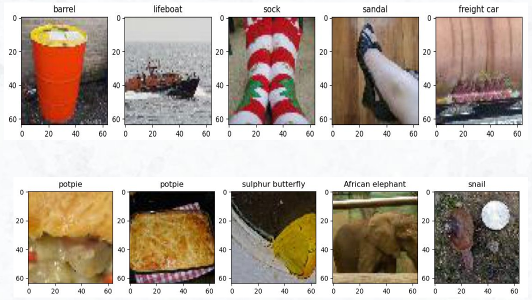


image (image)	label (class label)
	0 (n01443537)
	0 (n01443537)
	0 (n01443537)
	0 (n01443537)
	0 (n01443537)
	0 (n01443537)
	0 (n01443537)



Files

```
..
sample_data
tiny-imagenet-200
    test
        images
    train
    val
    wnids.txt
    words.txt
tiny-imagenet-200.zip
```

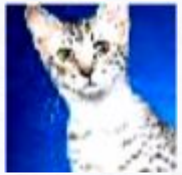


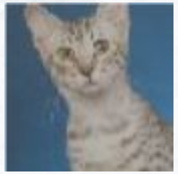
Tiny ImageNet Dataset

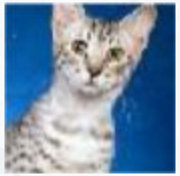
Data Augmentation

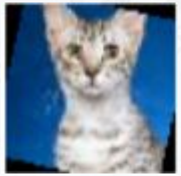
To prevent overfitting, we augment the data by horizontal flipping, translation and rotation.

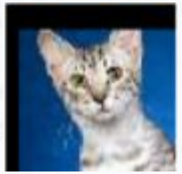
original picture


Contrast Adjustment 1.3


Contrast Adjustment 0.7


Horizontal Flipping


Random Rotation


Random Translation

PROJECT CHALLENGES

Increased Class Variability: the Tiny ImageNet dataset contains 200 diverse classes.

Model Size and Complexity: The resource constraints of low-power devices necessitate the development of compact models

Model Efficiency: achieving high accuracy while considering limited computational resources

3

MACHINE LEARNING MODELINGS

LINEAR MODEL

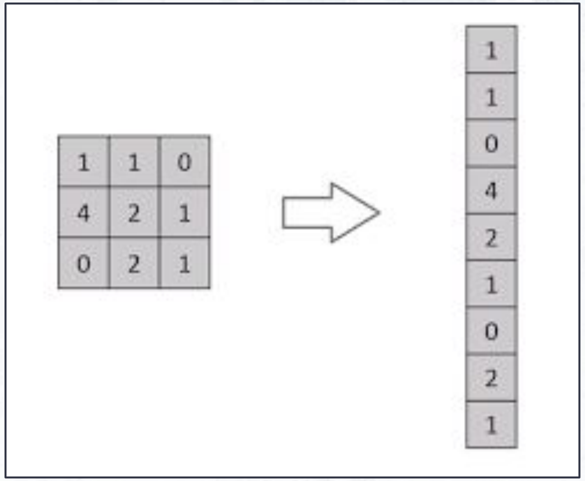
Linear models use only Linear transformations and work best for finding linear correlations in data.

They assume a direct relationship between two or more variables.

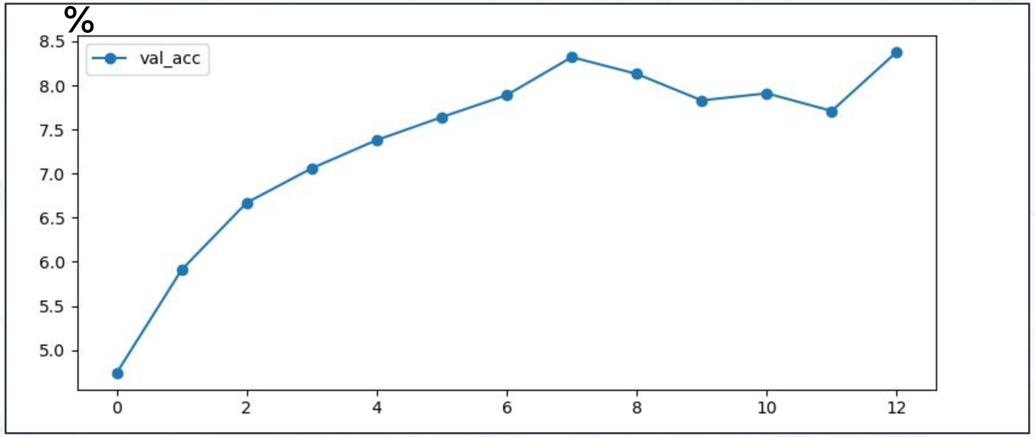
Areas of application:

- Finance
- Biology

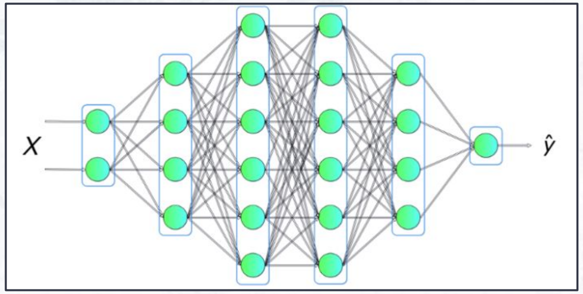
Typically not useful for image recognition.



LINEAR MODEL - Architecture and Validation



epoch



LINEAR MODEL - Conclusion

The basic Linear Model had a very low validation accuracy and a much higher training accuracy.

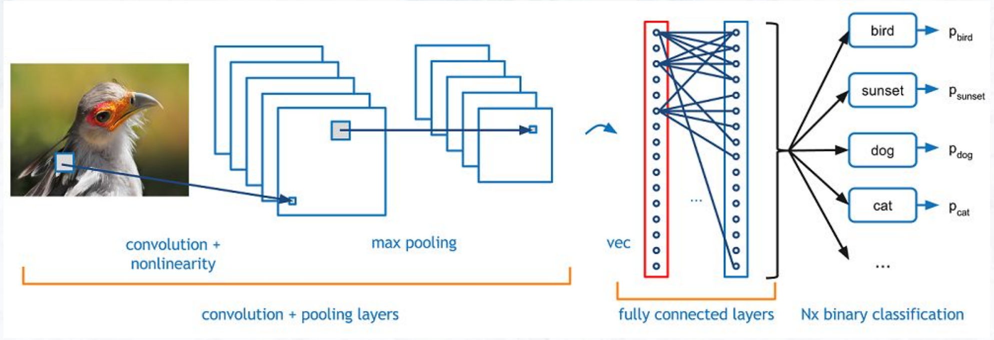
This means that the model was overfitting to the training data and not finding the correlations it needed to work on images outside of the training dataset.

- Final Training Accuracy: 44.9%
- Final Validation Accuracy: 8.38%

CNN

CNNs are specialized neural network architectures designed to process visual data by mimicking the organization of the human visual cortex.

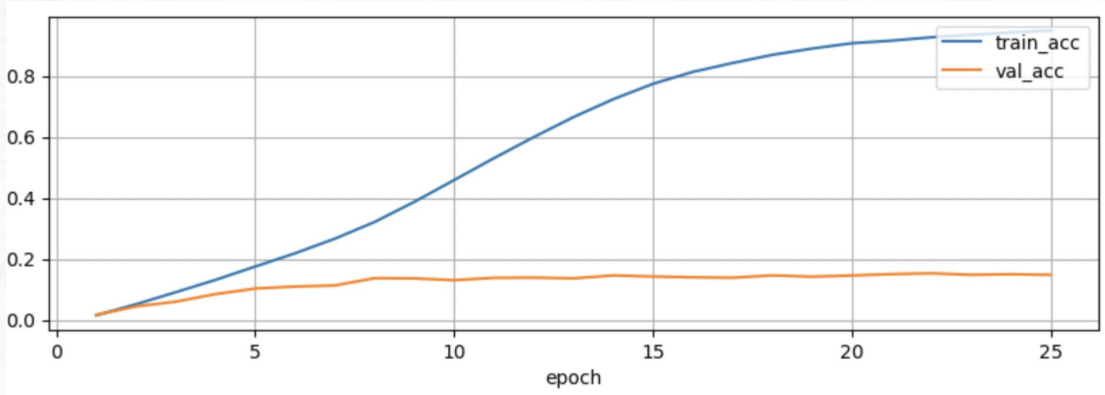
CNNs use convolutional layers to extract local features, pooling layers to downsample and aggregate information, and fully connected layers for classification.



2-LAYER CNN

We used a basic 2-layer CNN (kernel size=3, stride=1, padding=1) with maxpool and dropouts. Because of the convolutional layers' ability to spot features regardless of its position, the model can achieve high training accuracy. However, similar to linear model, basic CNN model also has the problem of overfitting.

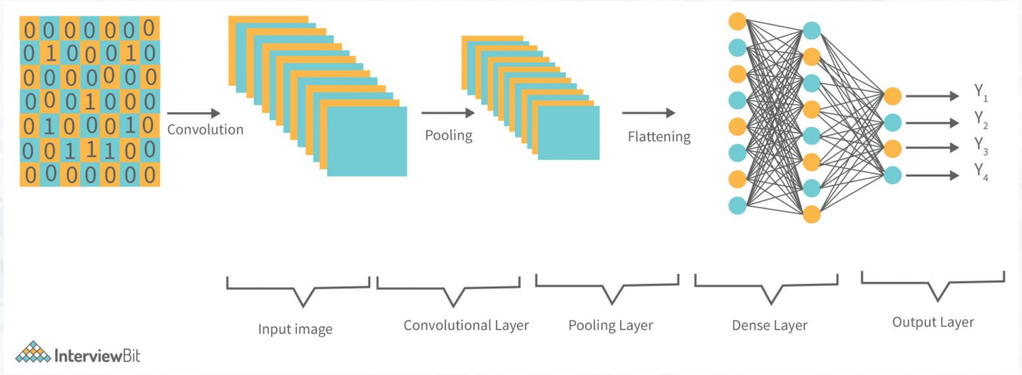
- Final Training Accuracy: 95.2%
- Final Validation Accuracy: 14.9%

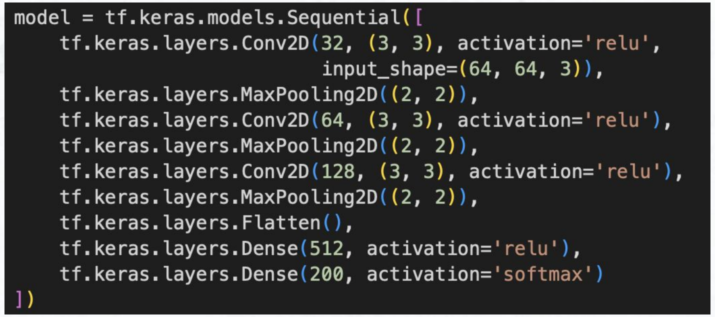


MULTI-LAYER CNN

We used a Multi-layer CNN with maxpool and dropouts

- Final Training Accuracy: 96.1%
- Final Validation Accuracy: 16.3%





```
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu',
                           input_shape=(64, 64, 3)),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Conv2D(128, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(200, activation='softmax')
])
```

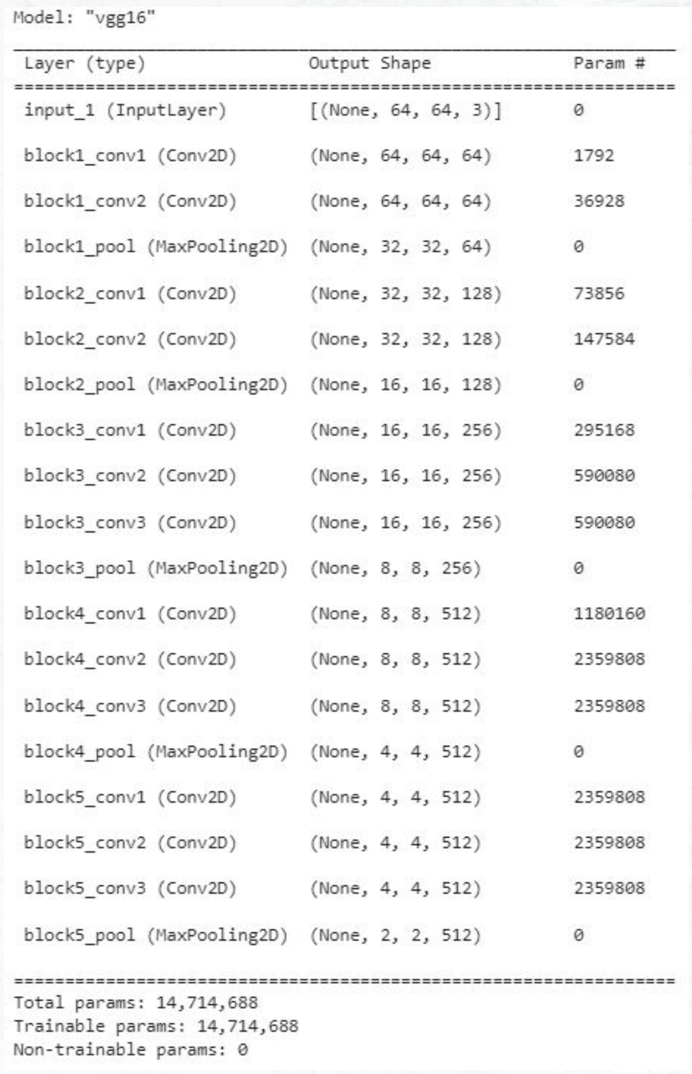
VGGNet

VGGNet, also known as the Visual Geometry Group Network, is a deep convolutional neural network (CNN) architecture

Simplicity, uniformity, deep structure

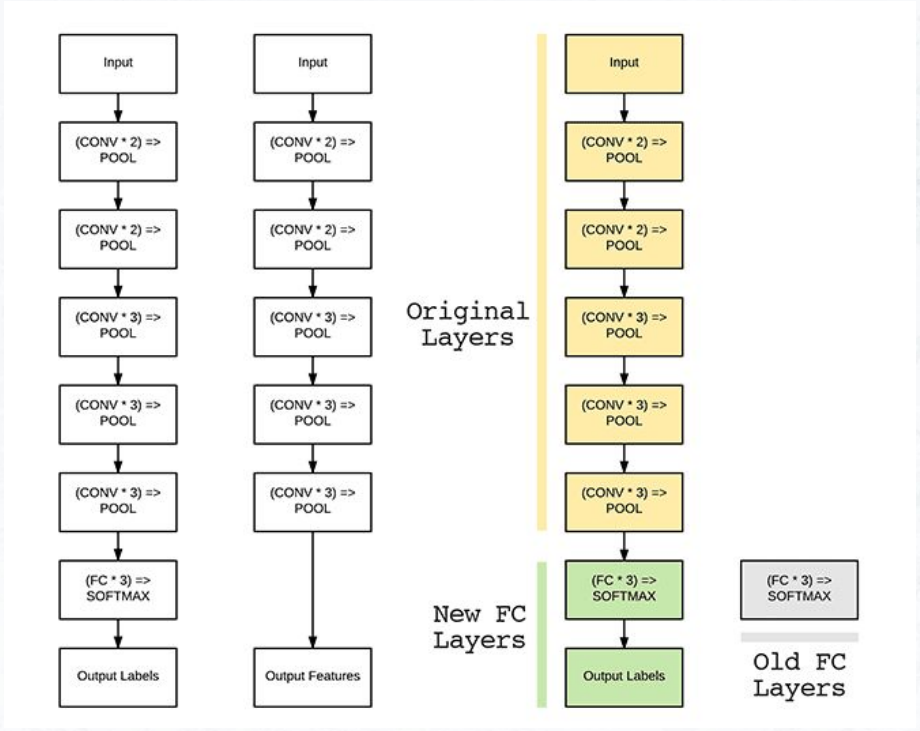
3x3 convolutional filters

VGGNet has several variations, denoted by the number of layers it contains. VGG16 and VGG19

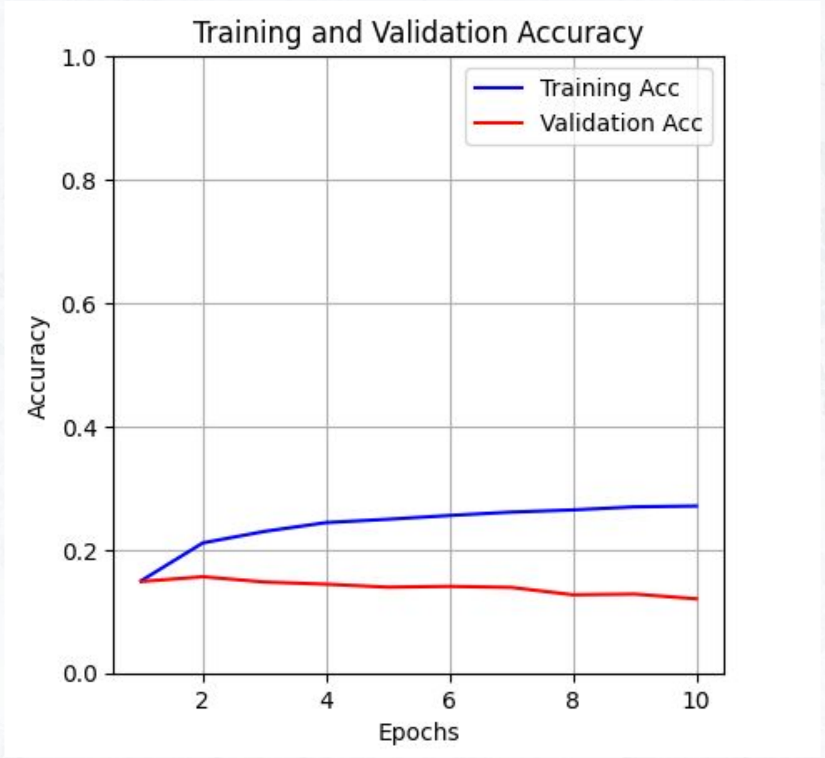


```
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 64, 64, 3)]       0

 block1_conv1 (Conv2D)       (None, 64, 64, 64)        1792

 block1_conv2 (Conv2D)       (None, 64, 64, 64)        36928

 block1_pool (MaxPooling2D)  (None, 32, 32, 64)        0

 block2_conv1 (Conv2D)       (None, 32, 32, 128)       73856

 block2_conv2 (Conv2D)       (None, 32, 32, 128)       147584

 block2_pool (MaxPooling2D)  (None, 16, 16, 128)       0

 block3_conv1 (Conv2D)       (None, 16, 16, 256)       295168

 block3_conv2 (Conv2D)       (None, 16, 16, 256)       590080

 block3_conv3 (Conv2D)       (None, 16, 16, 256)       590080

 block3_pool (MaxPooling2D)  (None, 8, 8, 256)         0

 block4_conv1 (Conv2D)       (None, 8, 8, 512)         1180160

 block4_conv2 (Conv2D)       (None, 8, 8, 512)         2359808

 block4_conv3 (Conv2D)       (None, 8, 8, 512)         2359808

 block4_pool (MaxPooling2D)  (None, 4, 4, 512)         0

 block5_conv1 (Conv2D)       (None, 4, 4, 512)         2359808

 block5_conv2 (Conv2D)       (None, 4, 4, 512)         2359808

 block5_conv3 (Conv2D)       (None, 4, 4, 512)         2359808

 block5_pool (MaxPooling2D)  (None, 2, 2, 512)         0

=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
```

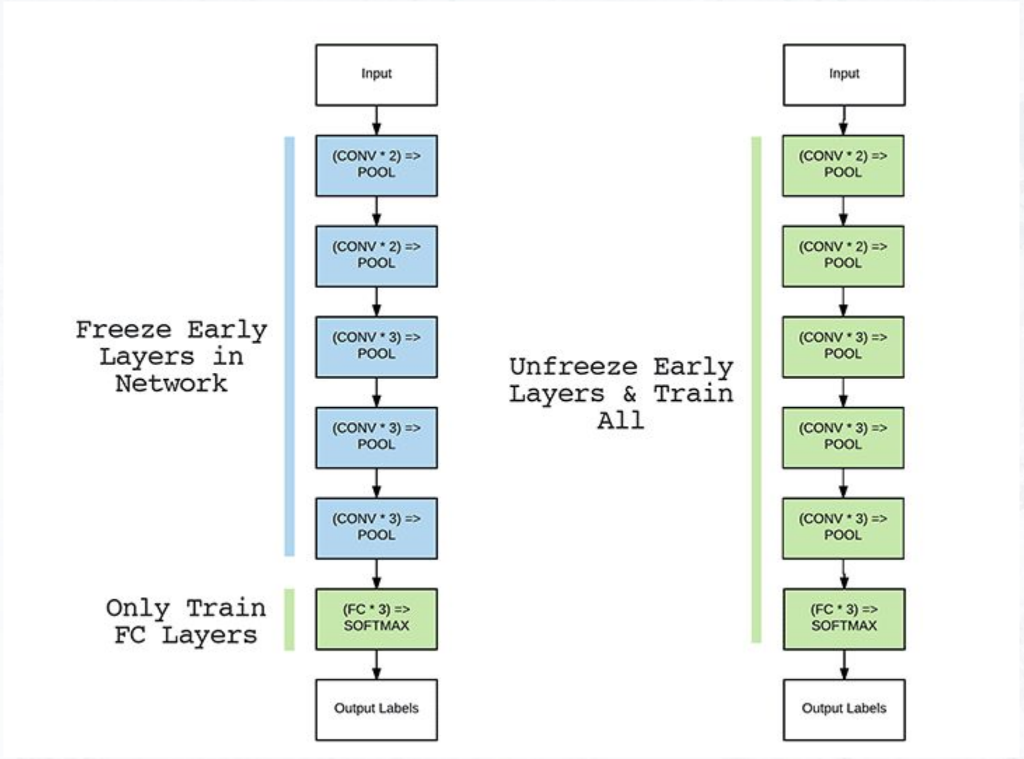
Transfer Learning



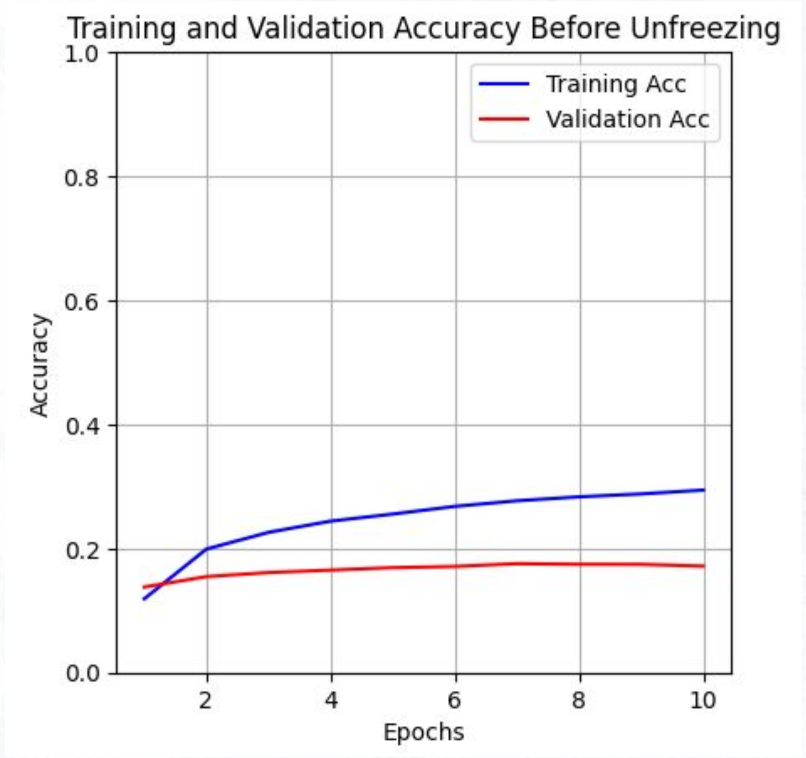
Transfer Learning

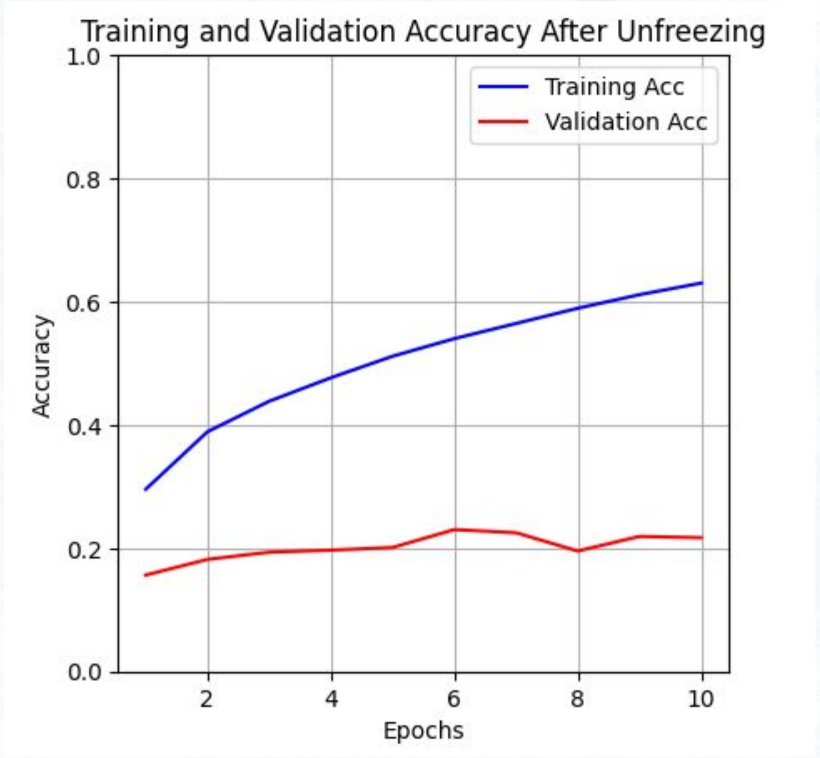


Layer-wise Fine Tuning



Layer-wise Fine Tuning

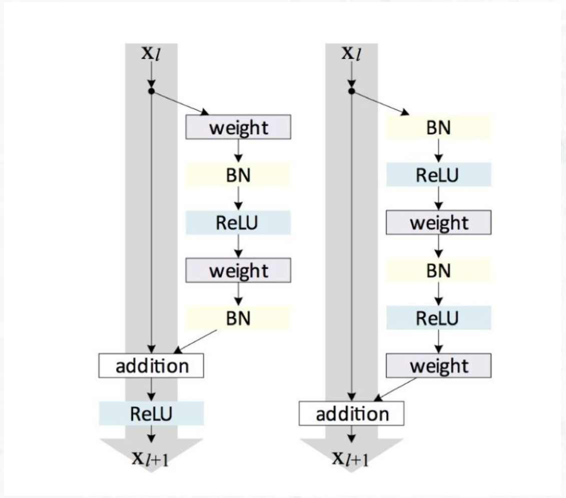


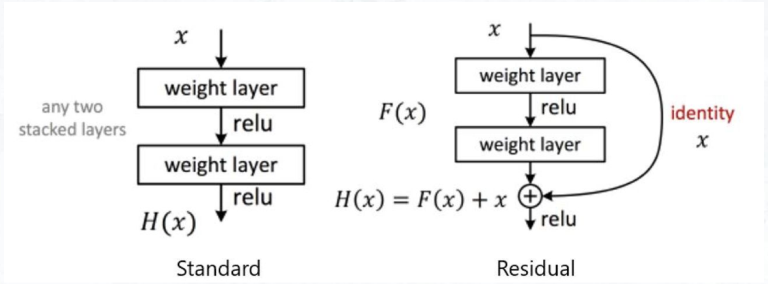


ResNet

ResNet, short for Residual Neural Network, is a groundbreaking deep learning architecture that revolutionized image classification tasks.

The key innovation lies in the residual blocks, which allow the network to learn residual mappings by directly adding skip connections between layers. These connections mitigate the vanishing gradient problem and facilitate the flow of information, enabling the network to effectively learn complex representations.

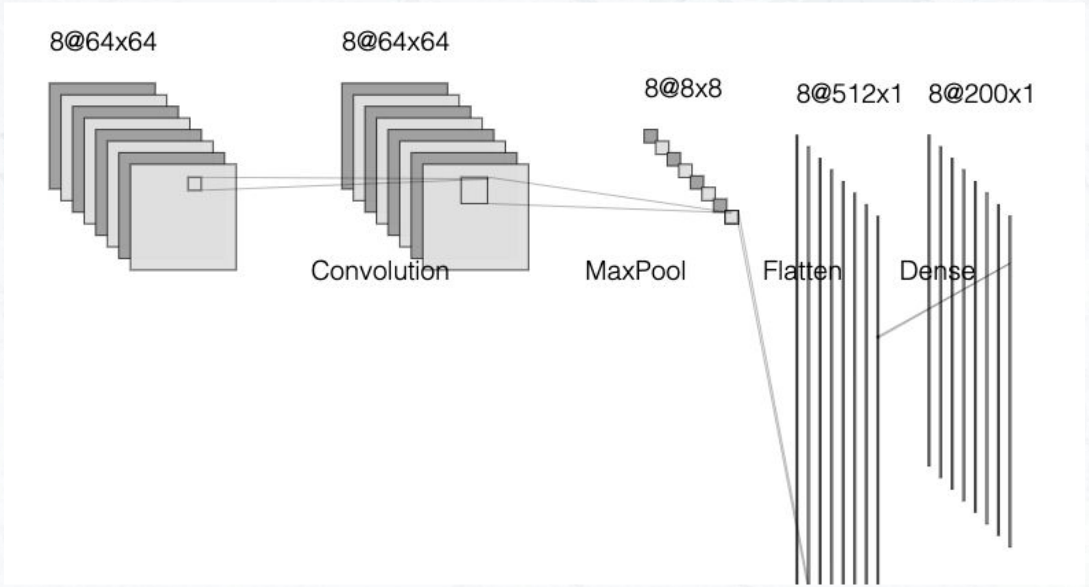




ResNet AND CNN

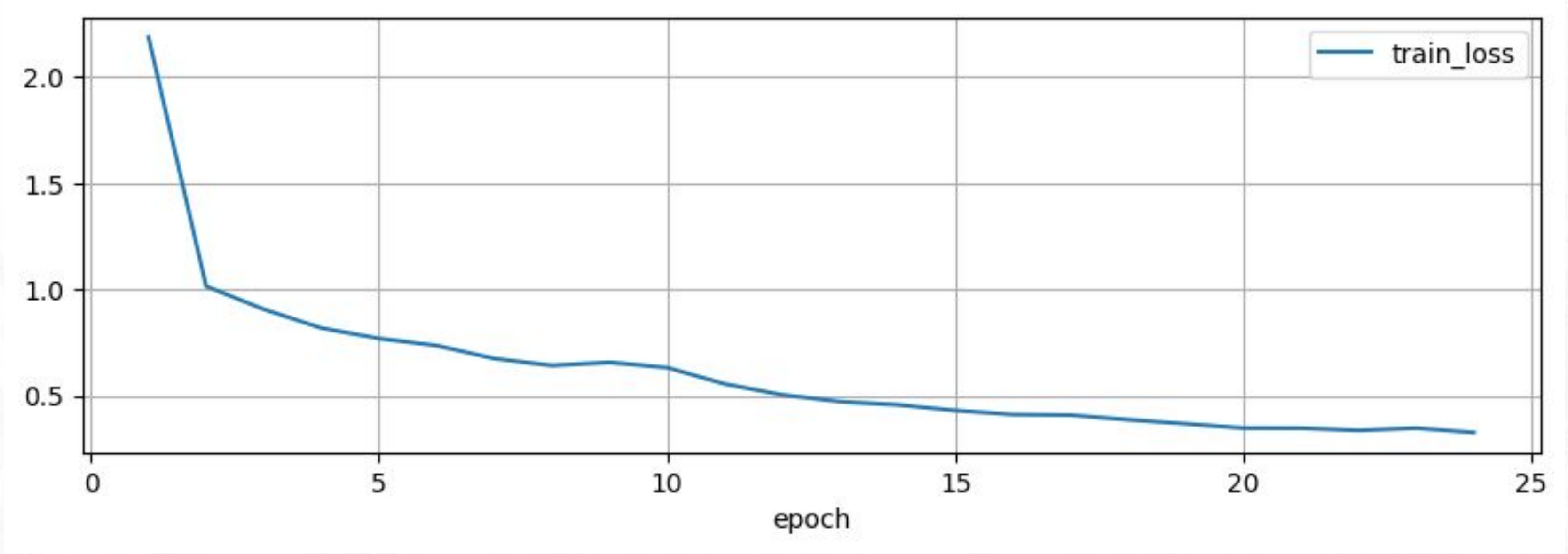
1. Convolutional layer
2. 2 Residual layer
3. Dense layer

Residual

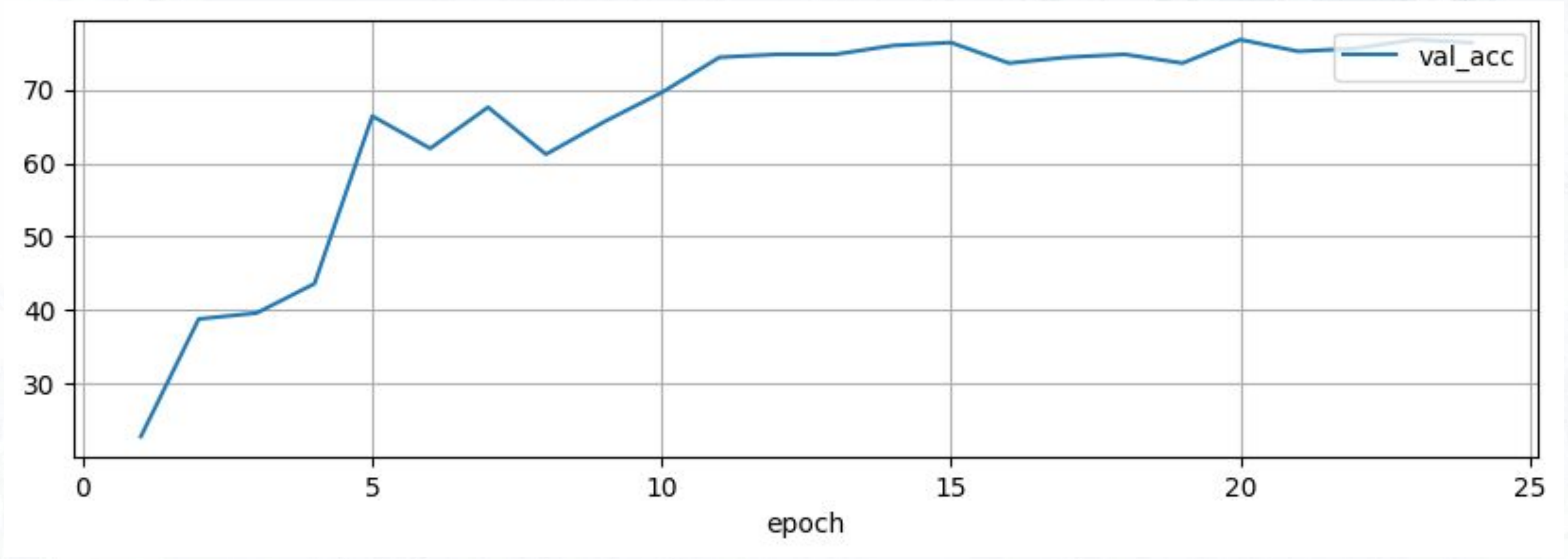


TRAIN LOSS

[ResNet + CNN]



[VALIDATION]ResNet + CNN

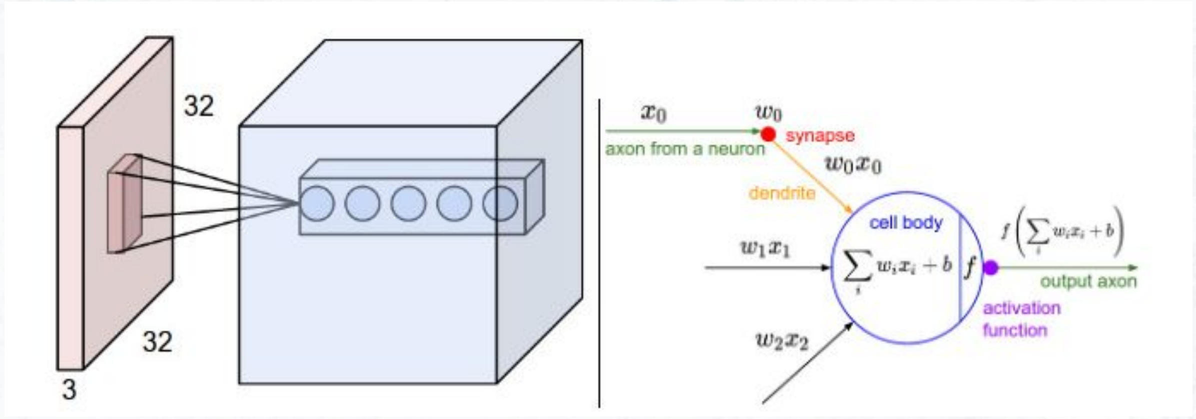


4

CONCLUSION AND DISCUSSION

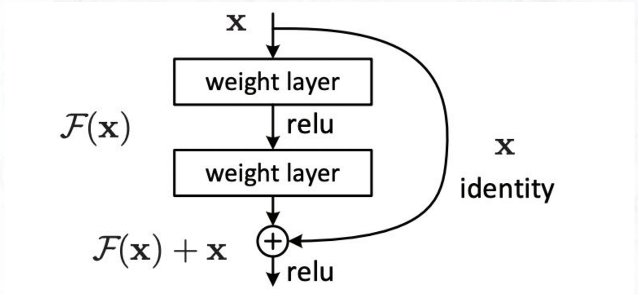
Why CNN Structure?

- Linear model only looks at pixel level
- Local Connectivity and weight sharing
- RGB image → 3 in_channels



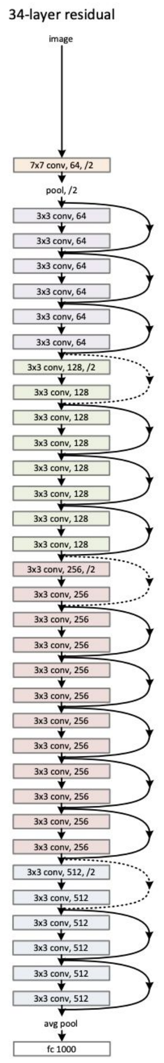
ResNet Performs Better Than Classic CNN

- More layers (152 vs 2)
- Skip connections
  - addresses the vanishing gradient problem
- Batch normalization
  - Stabilizes and regularizes the training process



x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 2. Residual learning: a building block.



5

CONTRIBUTIONS

OUR TEAM

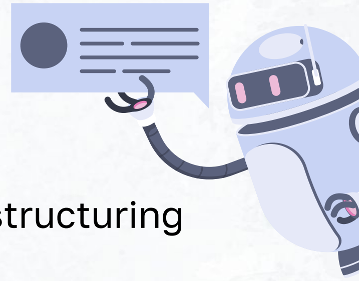
(-) Frank Dong
Responsible for slides structure, ResNet + CNN Model training, validation and report

(-) Nicholas Pavini
Responsible for Data Parsing and visualization, and Linear Model training / validation

(-) TianYun Yuan
Data Visualization, CNN and regular neural network comparison, VGGNet 16 training/validation, Data preprocessing

(-) Jingqi Yao
CNN model training, experimented and compare with ResNet

(-) Nima Hendi:
Responsible for Data Wrangling, Multi-Layer CNN Model, Validation and Restructuring of slides and the report



Reference

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. Pages 1097–1105, 2012.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167, 2015.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1):1929–1958, 2014.