Freshness of Fruits & Vegetables

- Chuheng Yu
- Kwan Wing Tuet
- Tyler Christoforo
- Weiming Wang

Objectives

- Improve consumer's awareness of food freshness to reduce waste
  - Grocery shopping
  - Management of food in fridge

- Dataset:
  - 7000 fruit & vegetable images
  - 10 different kinds
  - Fresh or not for each kind
  - www.mdpi.com/1424-8220/22/21/8192

Preprocessing

- Split the data into train, val, test folders
- Divide into 20 classes (10 fruits and veges * fresh or not) - 5% threshold
- Create data generator for data augamentation

```
Found 6966 images belonging to 20 classes.
Found 2313 images belonging to 20 classes.
Found 2335 images belonging to 20 classes.
```

```
Widths: Mode = 224, Average = 340.24, Median = 224.0
Heights: Mode = 224, Average = 293.81, Median = 224.0
```

```
FreshApple: 367
FreshBanana: 372
FreshBellpepper: 366
FreshCarrot: 370
FreshCucumber: 248
FreshMango: 363
FreshOrange: 365
FreshPotato: 369
FreshStrawberry: 361
FreshTomato: 360
RottenApple: 349
RottenBanana: 344
RottenBellpepper: 315
RottenCarrot: 348
RottenCucumber: 329
RottenMango: 360
RottenOrange: 354
RottenPotato: 318
RottenStrawberry: 355
RottenTomato: 353
```

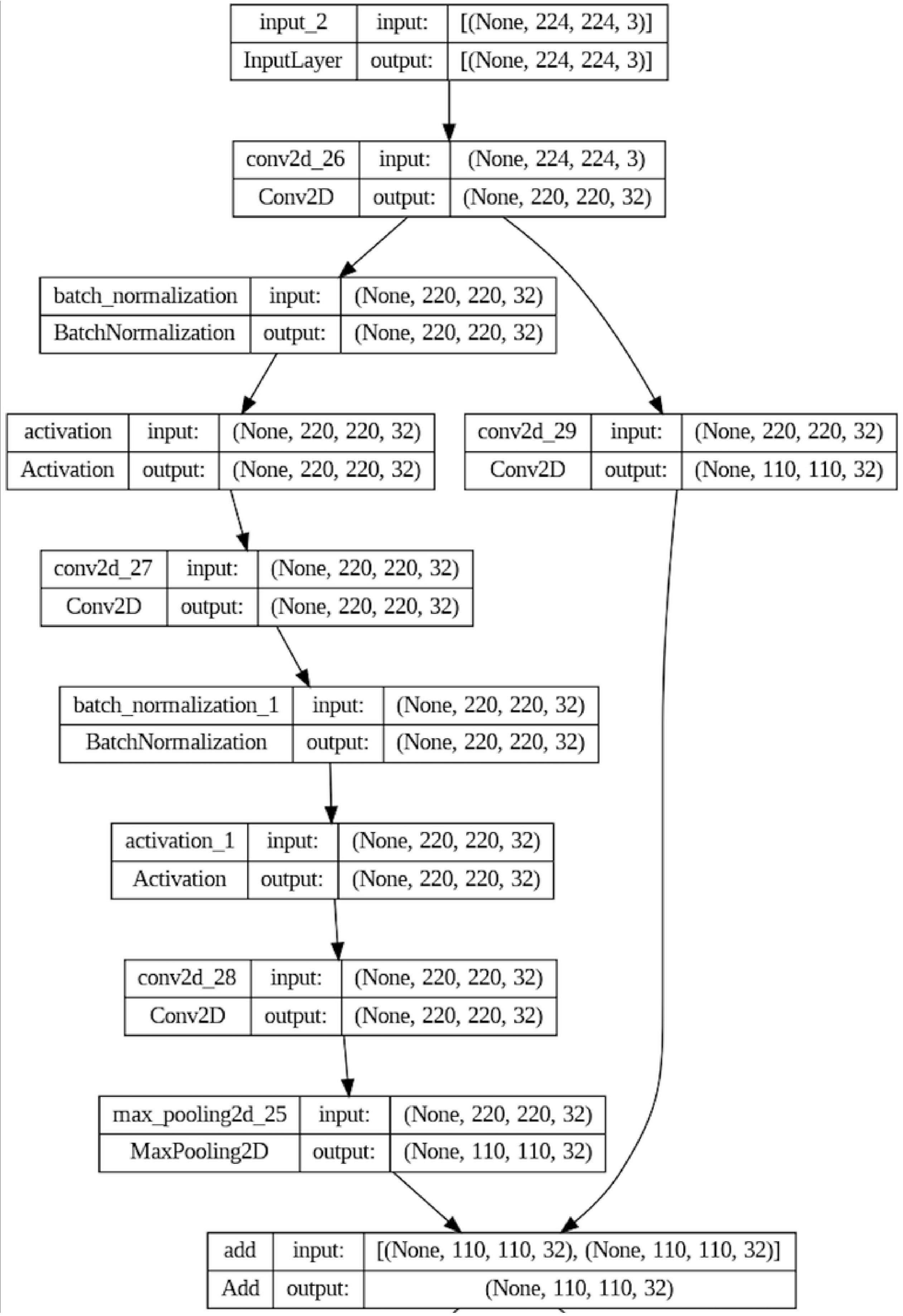
Sequential CNN

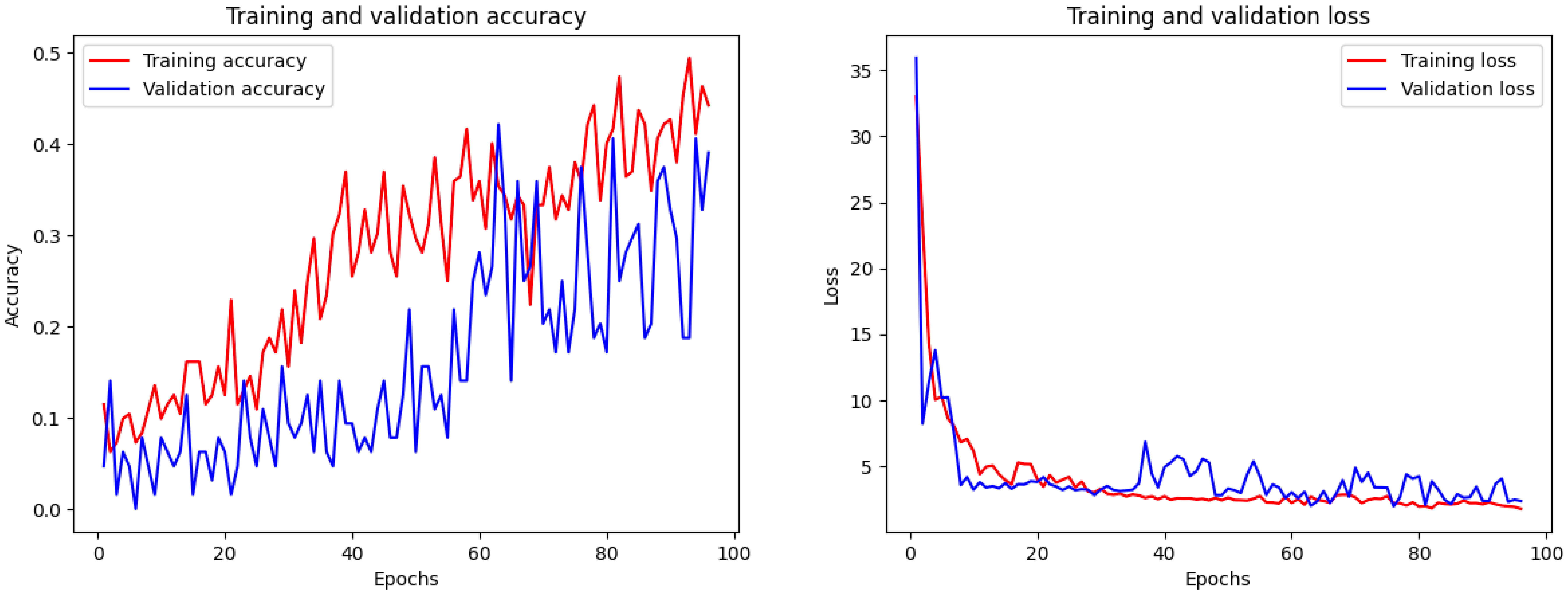
- 16 layers
  - 6 Conv2D layers
  - filters increase from 32 to 1024
  - 3 x 3 grid to detect feature
- Each Conv2D follow by a 2 x 2 MaxPooling layer
- Activation function mixture of softmax, relu, and tanh
- Performance:
  - loss: 2.9965
  - accuracy: 0.0833
  - val_loss: 2.9841
  - val_accuracy: 0.1094

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_20_input (InputLayer  [(None, 224, 224, 3)]    0
 )

 conv2d_20 (Conv2D)          (None, 222, 222, 32)      896

 max_pooling2d_19 (MaxPoolin  (None, 111, 111, 32)     0
 g2D)

 conv2d_21 (Conv2D)          (None, 109, 109, 64)      18496

 max_pooling2d_20 (MaxPoolin  (None, 54, 54, 64)       0
 g2D)

 conv2d_22 (Conv2D)          (None, 52, 52, 128)       73856

 max_pooling2d_21 (MaxPoolin  (None, 26, 26, 128)      0
 g2D)

 conv2d_23 (Conv2D)          (None, 24, 24, 256)       295168

 max_pooling2d_22 (MaxPoolin  (None, 12, 12, 256)      0
 g2D)

 conv2d_24 (Conv2D)          (None, 10, 10, 512)       1180160

 max_pooling2d_23 (MaxPoolin  (None, 5, 5, 512)        0
 g2D)

 conv2d_25 (Conv2D)          (None, 5, 5, 1024)        4719616

 max_pooling2d_24 (MaxPoolin  (None, 2, 2, 1024)       0
 g2D)

 flatten_2 (Flatten)         (None, 4096)              0

 dense_1 (Dense)             (None, 128)               524416

 output (Dense)              (None, 20)                2580

=================================================================
Total params: 6,815,188
Trainable params: 6,815,188
Non-trainable params: 0
_________________________________________________________________
```

CNN with Residual

- 59 total layers
  - 19 Conv2D layers
  - Filters increase from 32 to 1024
  - 3 x 3 grid to detect feature
- Batch Normalization layer follow by a Conv2D
- Relu and softmax
- Maxpooling: 3x3, strides = 2
- Performance:
  - loss: 2.6769
  - accuracy: 0.3542
  - val_loss: 1.9757
  - val_accuracy: 0.4219

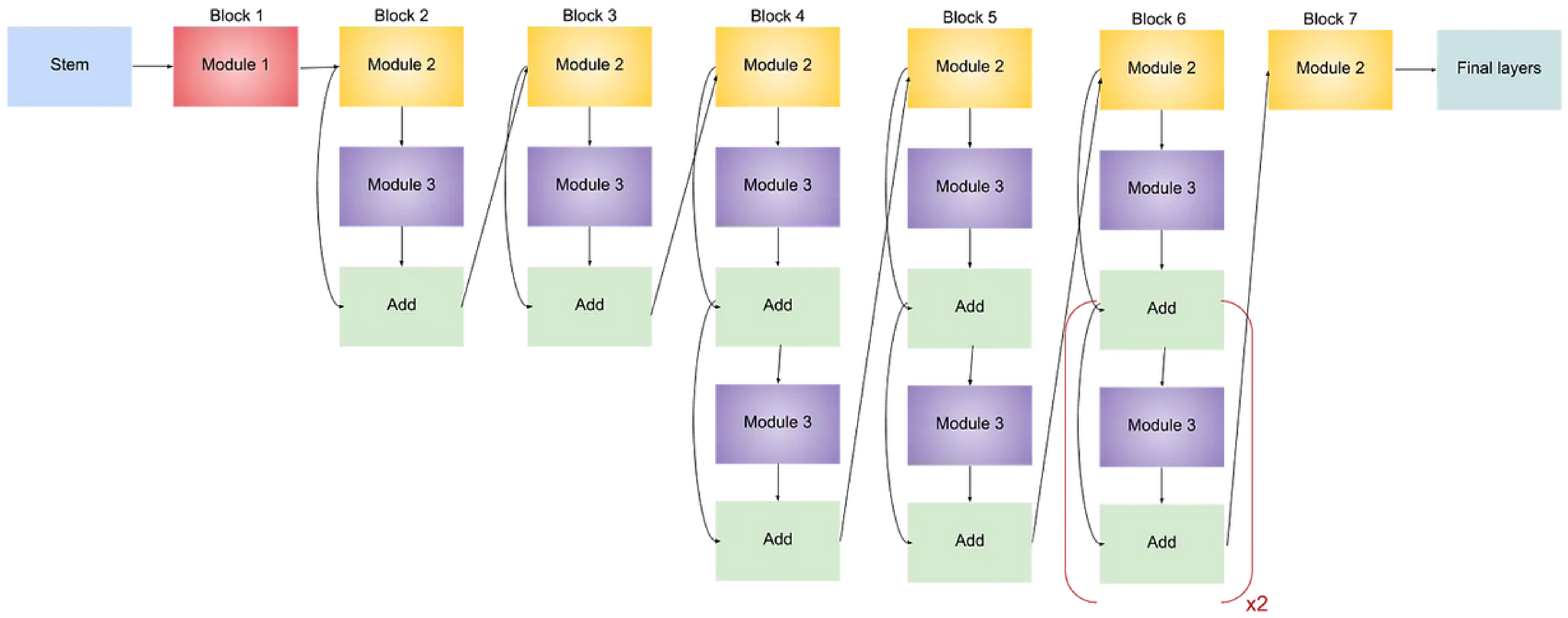




Training and validation accuracy
Training accuracy
Validation accuracy
Accuracy
Epochs
0.0
0.1
0.2
0.3
0.4
0.5
0
20
40
60
80
100
Training and validation loss
Training loss
Validation loss
Loss
Epochs
5
10
15
20
25
30
35
0
20
40
60
80
100

EfficientNetB0

- 237 layers
- Sub-blocks model including:
  - Conv2D, Zero Padding, Batch Normalization, Rescaling
- Performance:
  - Set patience of early stopper: 30
  - Epoch: 200
  - loss: 2.9727
  - accuracy: 0.1094
  - val_loss: 3.0069
  - val_accuracy: 0.0781



ResNet50

Background

- Originally named Residual net in 2015
- One of the most popular models
- Achieved a top-5 error rate at around 5%

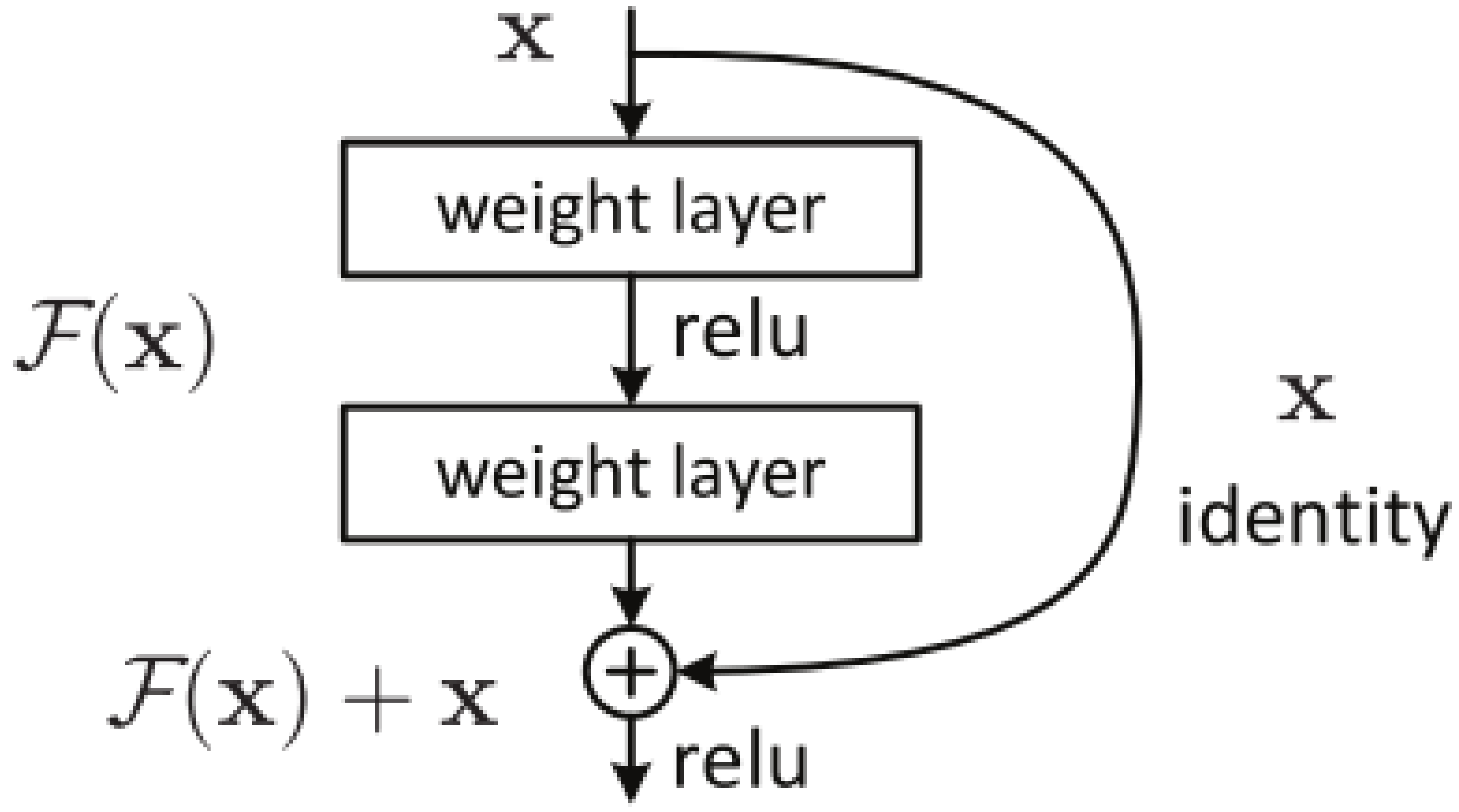
Specialty

- Aimed to tackle vanishing gradient descent
- Bypass/skip the layers in between layers
- "Fitting a residual mapping is easier"

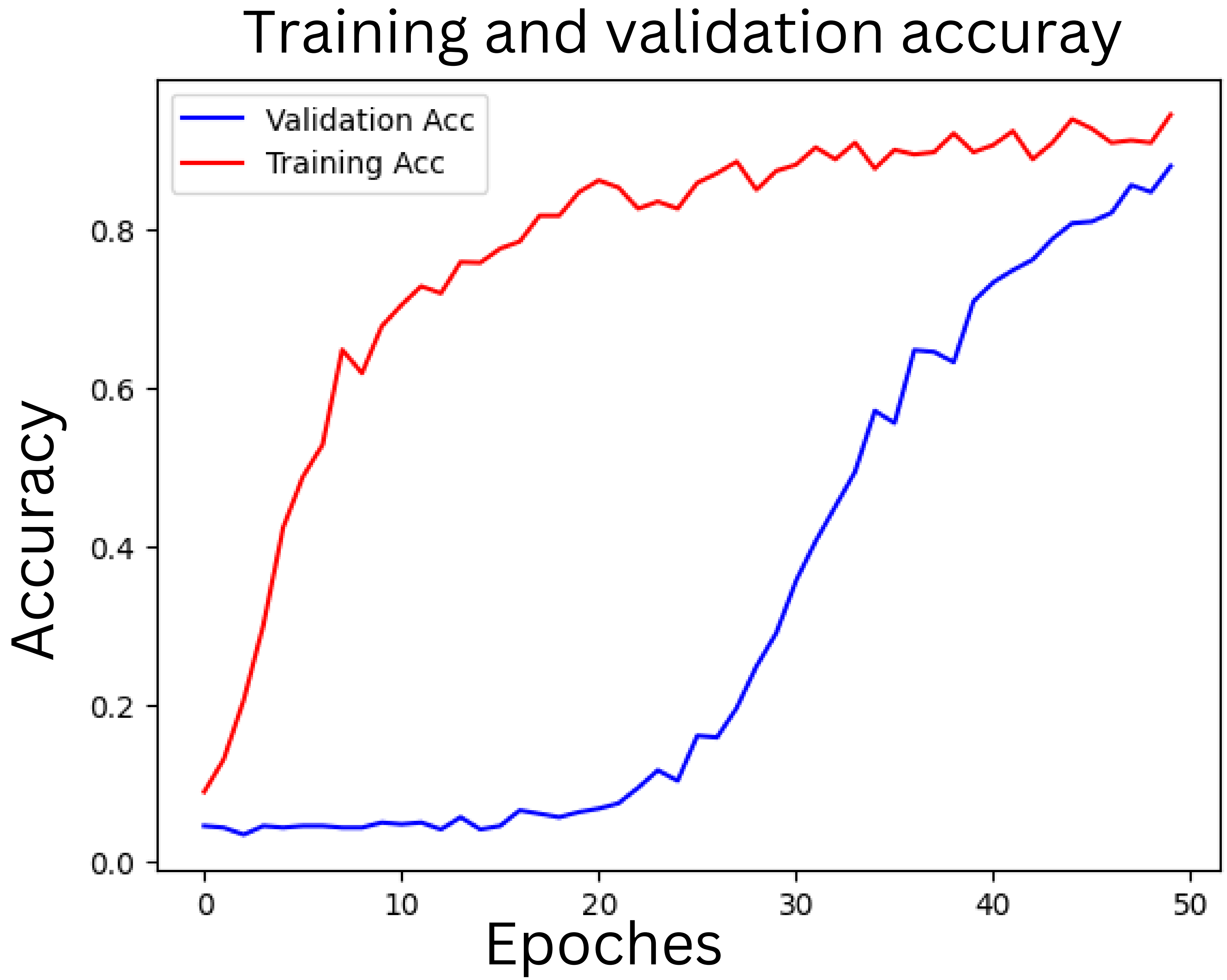
50 layers

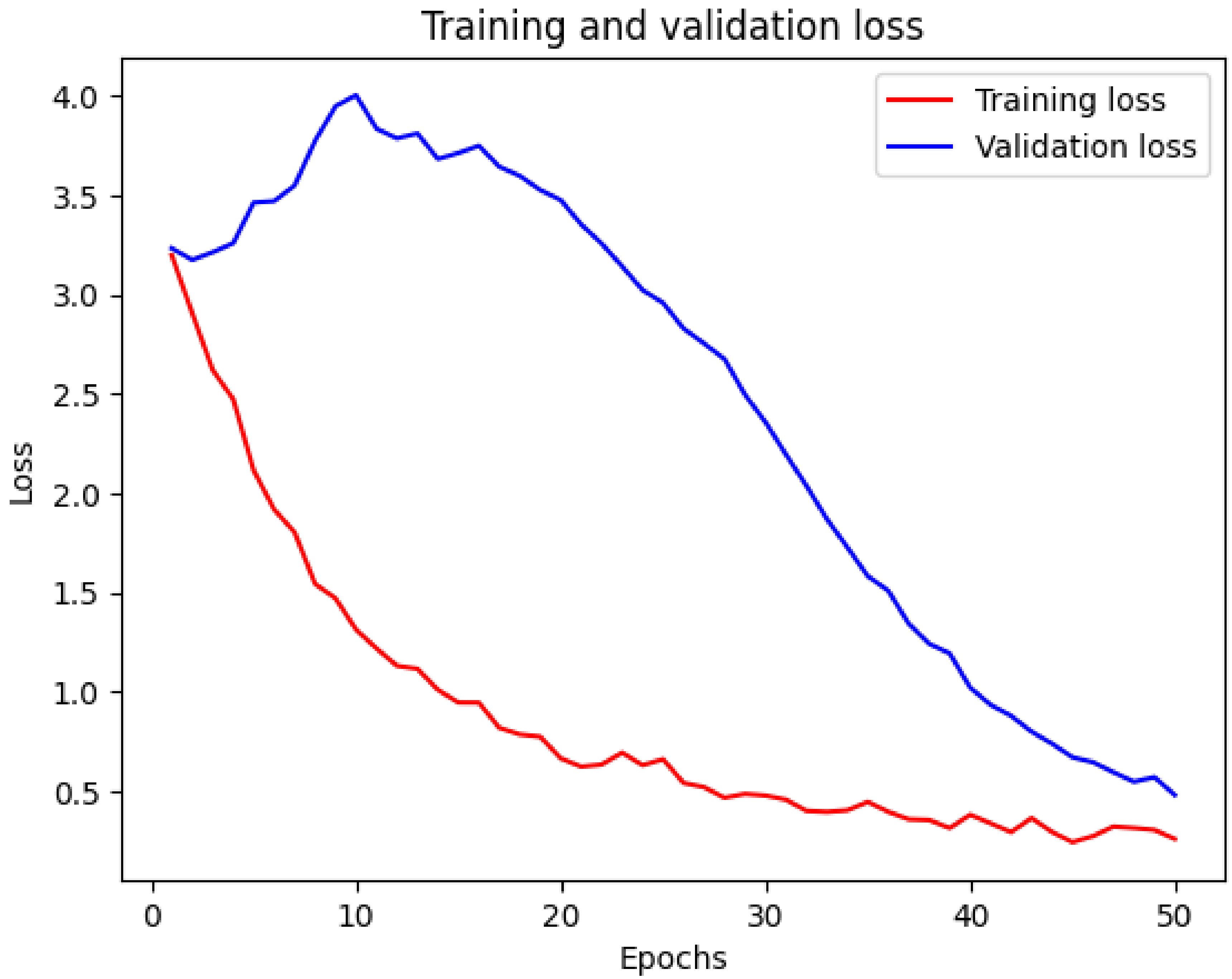
- **48 Convolution layer**
- **1 MaxPool layer**
- **1 average pool layer**

Identity short cut connection



Performance Review

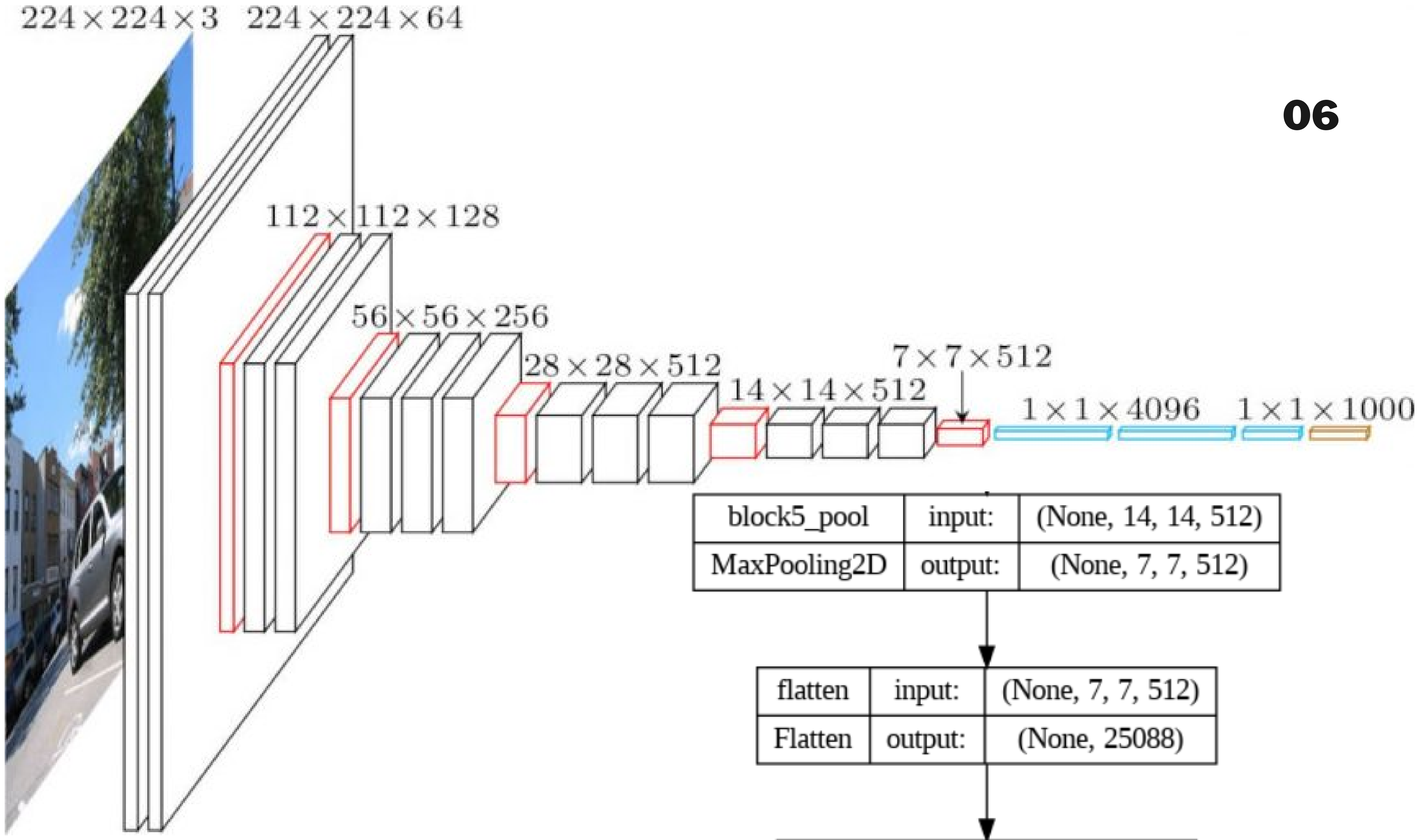


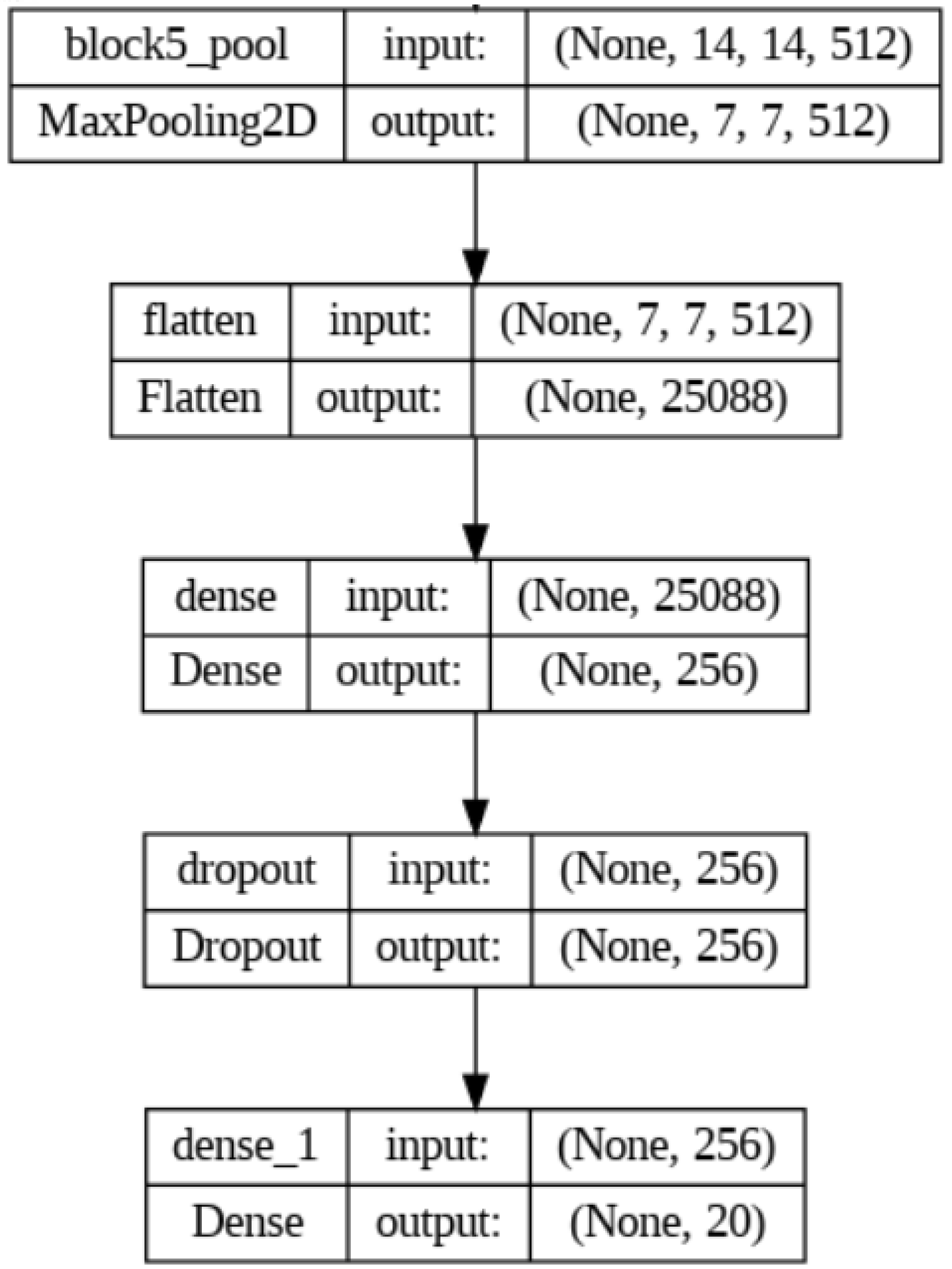


- Loss function: categorical_crossentropy
- Given enough epoches it could reach 90%

VGG-16

- Introduced in 2014 by a team from "Visual Geometry Group" at Oxford
- Has only 16 layers, including convolutional layers, pooling layers, and fully connected layers
- trained on the large-scale ImageNet dataset





block5_pool	input:	(None, 14, 14, 512)
MaxPooling2D	output:	(None, 7, 7, 512)

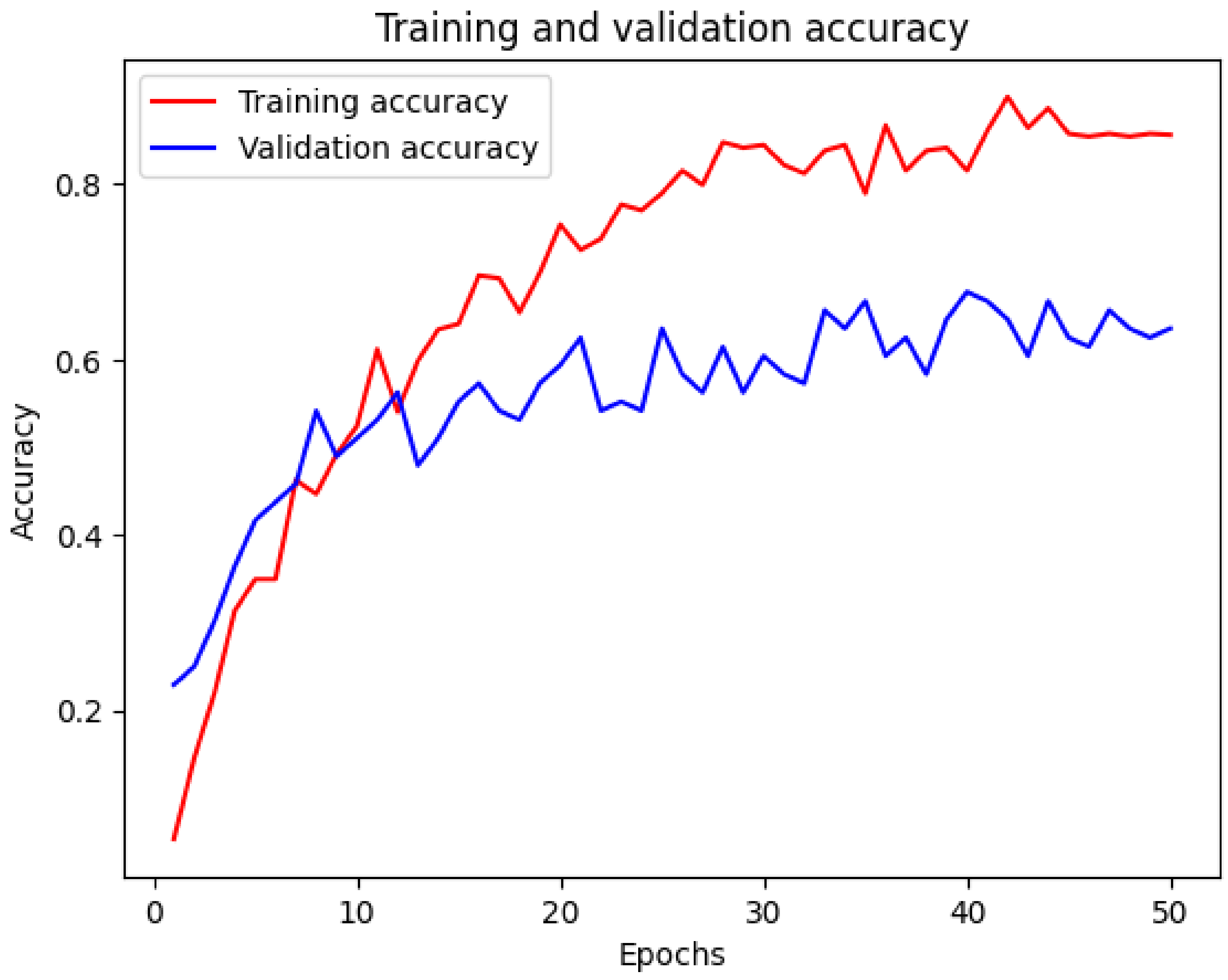
flatten	input:	(None, 7, 7, 512)
Flatten	output:	(None, 25088)

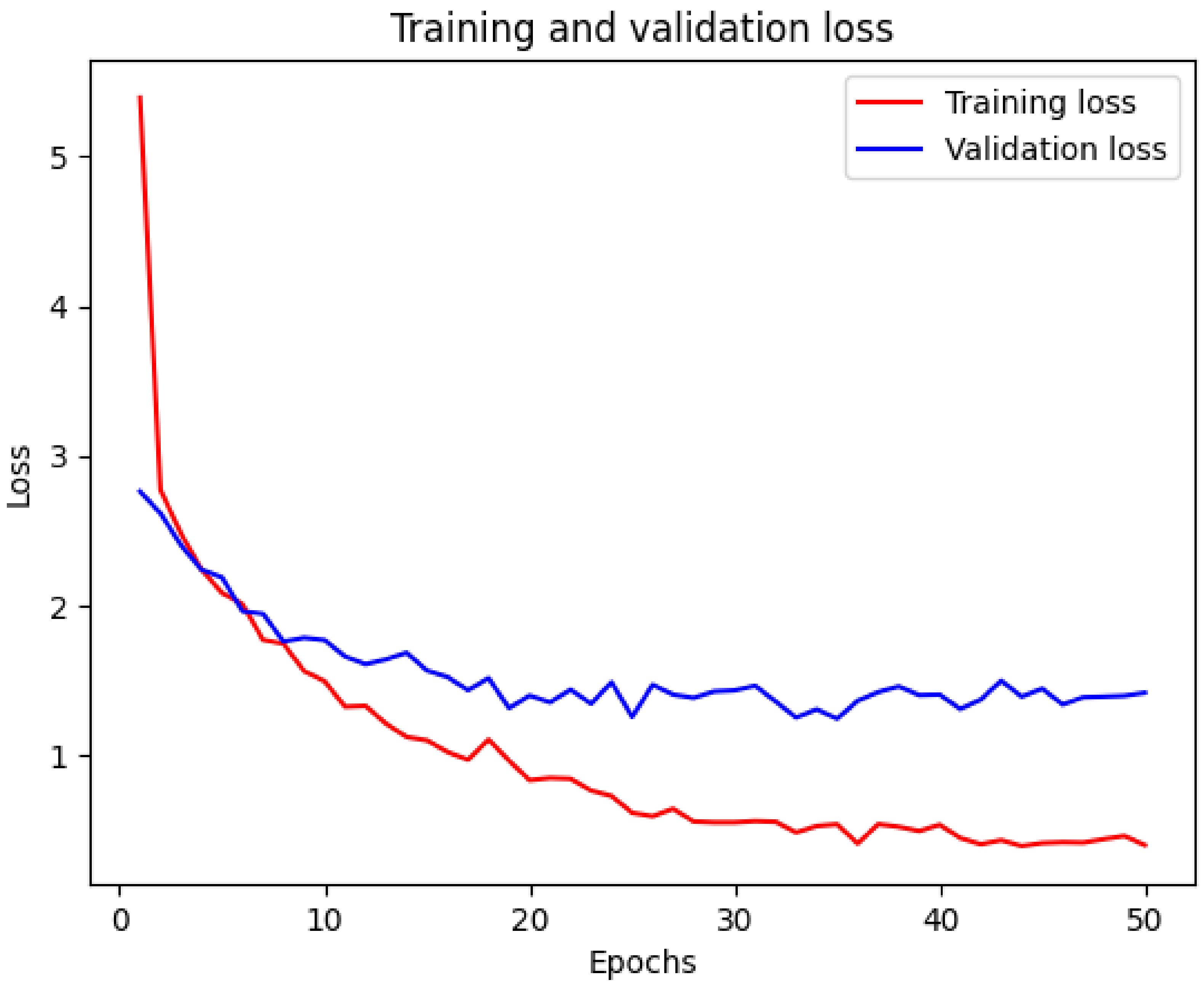
dense	input:	(None, 25088)
Dense	output:	(None, 256)

dropout	input:	(None, 256)
Dropout	output:	(None, 256)

dense_1	input:	(None, 256)
Dense	output:	(None, 20)

Performance Review





- Early Stopping Patience = 15
- Validation Accuracy: 0.6771
- Only took 5% as sample due to runtime limitation

VGG-16
Version2

```
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 224, 224, 3)]     0

sequential (Sequential)      (None, 224, 224, 3)       0

tf.__operators__.getitem (S  (None, 224, 224, 3)       0
licingOpLambda)

tf.nn.bias_add (TFOpLambda)  (None, 224, 224, 3)       0

vgg16 (Functional)           (None, None, None, 512)   14714688

flatten (Flatten)            (None, 25088)             0

dense (Dense)                (None, 256)               6422784

dense_1 (Dense)              (None, 20)                5140

=================================================================
Total params: 21,142,612
Trainable params: 6,427,924
Non-trainable params: 14,714,688
```

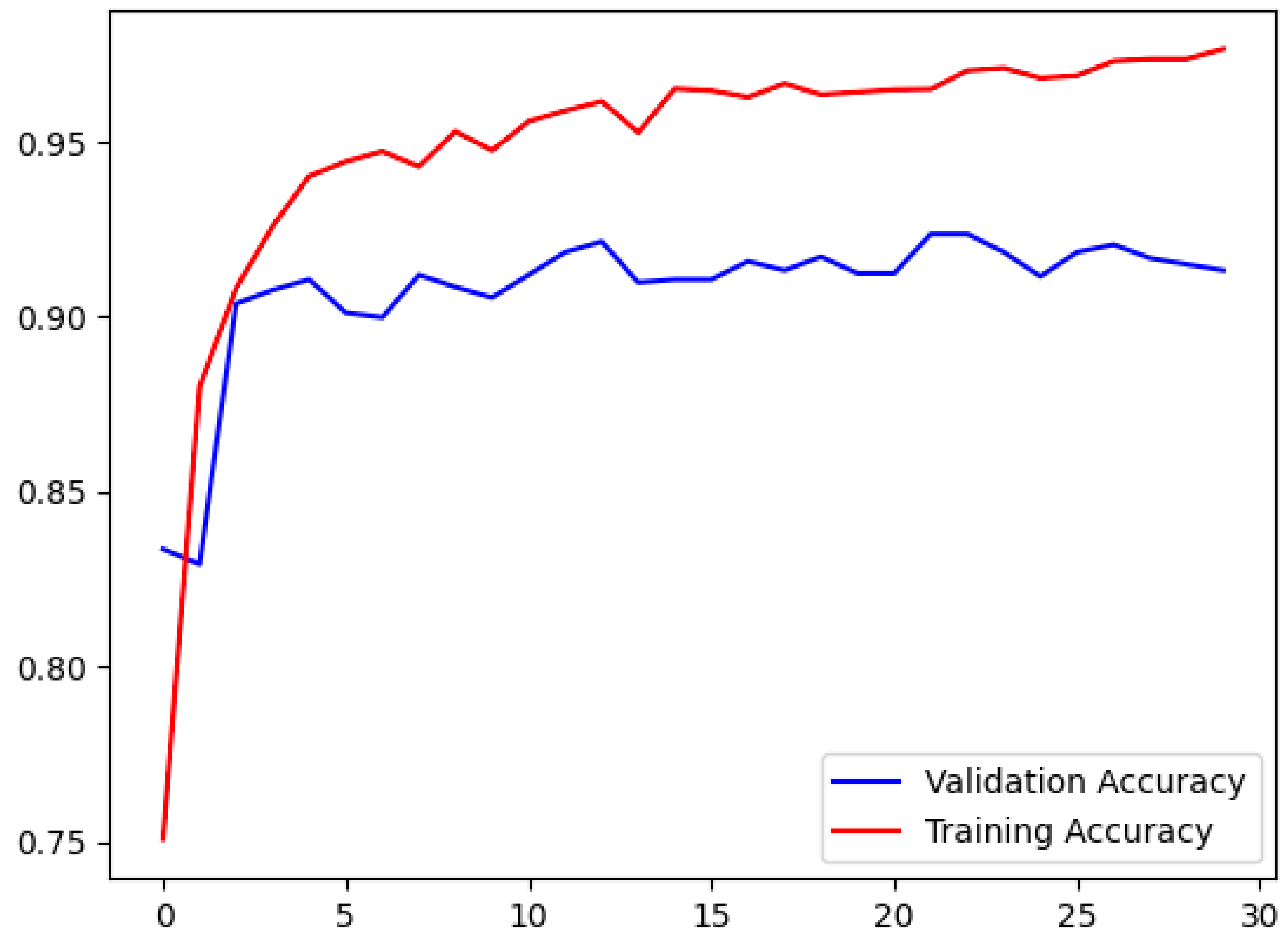
- **Data Augmentation layer (*SlicingOpLambda*)**
- **Preprocessing_input layer (*TFOpLambda*)**

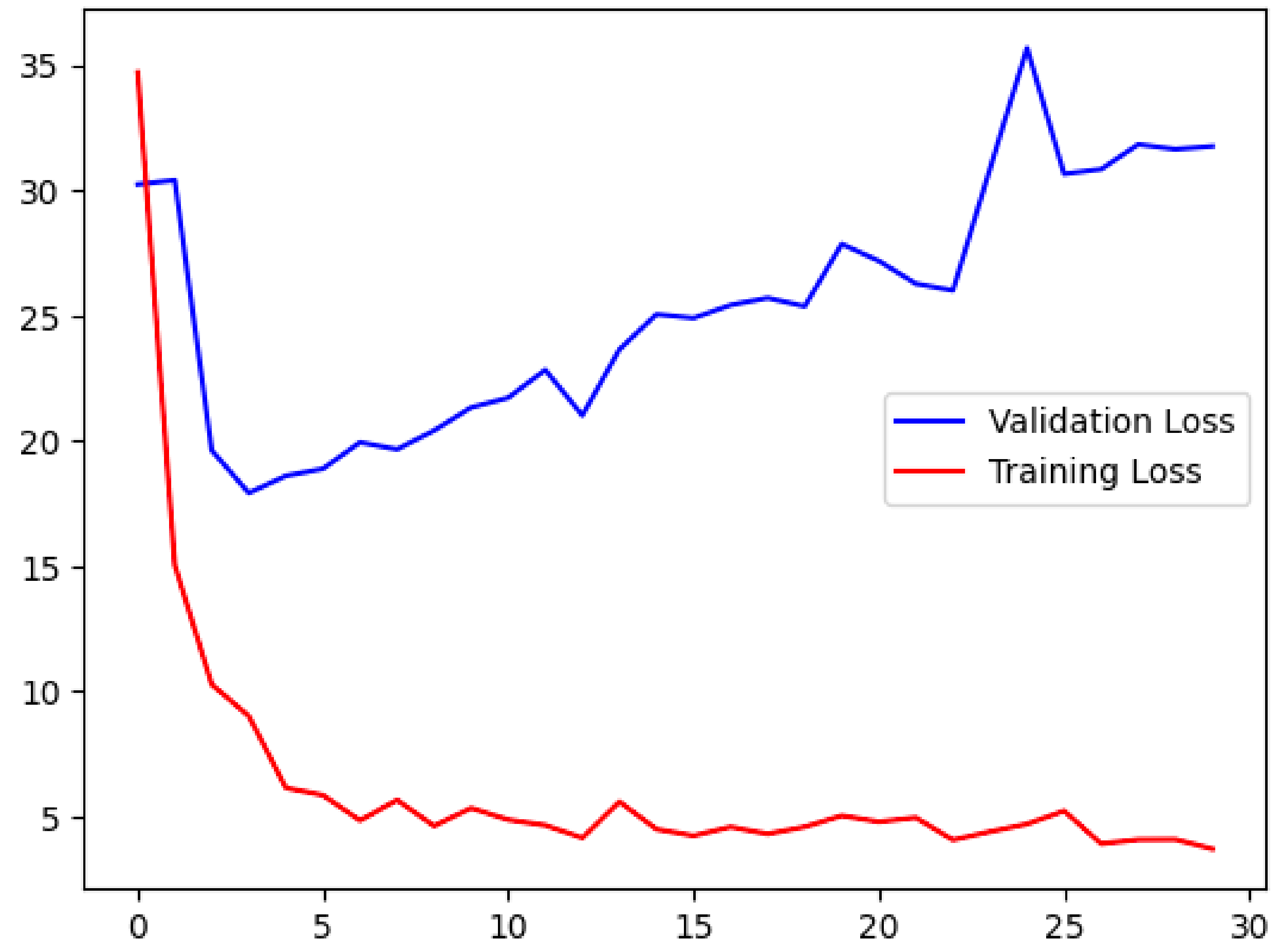
```
inputs = keras.Input(shape=(224, 224, 3))
x = data_augmentation(inputs)
x = keras.applications.vgg16.preprocess_input(x)
x = conv_base(x)
x = layers.Flatten()(x)
x = layers.Dense(256)(x)
outputs = layers.Dense(20, activation="softmax")(x)
model = keras.Model(inputs=inputs, outputs=outputs)
```

```
model.compile(optimizer="adam",
              loss="sparse_categorical_crossentropy",
              metrics=['accuracy'])
```

- **Loss = "sparse_categorical_crossentropy"**

Performance Review





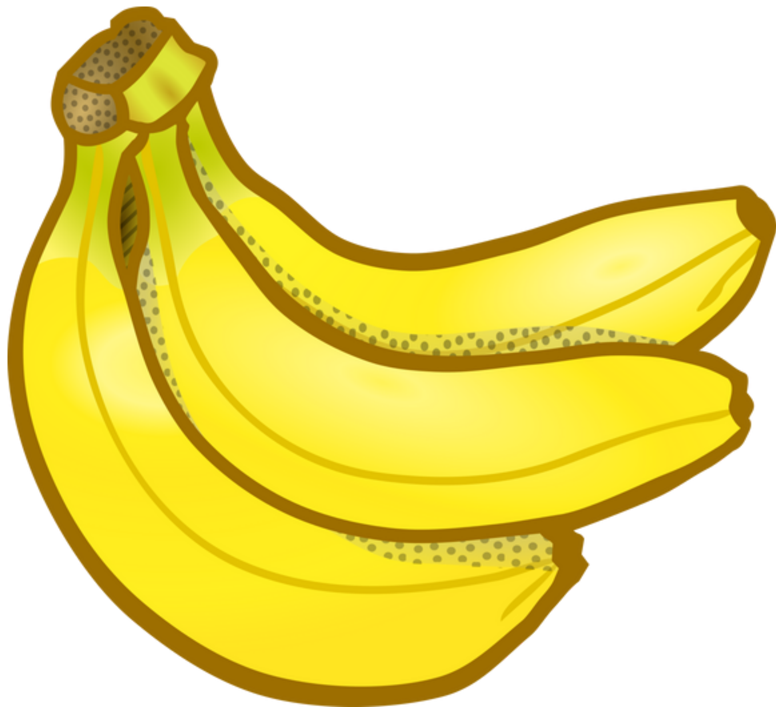
- **Validation Accuracy: plateau after 5th epoch**
- **Val Loss: Gradually Increasing**

- **Epoch: 30**
- **BatchSize: 32**

Testing & closing thoughts

```
Accuracy: 93.23%
```

- Image unification
  - File formatting issue (webpage extension vs. jpg)
- Include more data for training
  - The dataset only had 7000 images across 20 labels
- More filters for CNN
  - Regularization to reduce loss; Learning rate to increase performance
- Modify pre-train model
  - Add or modify layers based on more research



Thank you!