Oanh Le

CIS4930 Deep Learning Fundamentals

Project 1-2 Report

Data

Number of classes: 11
Training data set has 7213 images
Validation data set has 1543 images

Training parameters

Batch size: 128
Image size: (256, 256)
Learning rate: 0.001
Number of epochs: 100
Drop out: 50%
Patience: 10

Data augmentation

I used *ImageDataGenerator* to augment the images while training my model. This technique helps to prevent overfitting and to generalize the model better. There is total of three transformations added:

- Shear the image by 20%
- Zoom 20% in on the image
- Horizontal flip

Model summary

1. Keras Pretrained Model
   I used VGG16 architecture to pretrain my model. I set include_top = false to exclude the fully-connected layer and only convolutional and pooling layers are imported. The pretrained model summary is showed as below.

```
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 256, 256, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 256, 256, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 256, 256, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 128, 128, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 128, 128, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 128, 128, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 64, 64, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 64, 64, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 64, 64, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 64, 64, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 32, 32, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 32, 32, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 32, 32, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 32, 32, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 16, 16, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 16, 16, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 16, 16, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 16, 16, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 8, 8, 512)         0
=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
```

2. Sequential model

   I added my own layers after pretraining the model. The below model was the one with the best accuracy among different attempts:

   Convolution layers:

   → 1 x convolution layer of 512 channel of 3x3 kernel

   → 1 x convolution layer of 1024 channel of 3x3 kernel

   → 1 x maxpool layer of 2x2 pool size

   RELU activation is added to each layer. After creating the convolution layers, I flatten the vector and pass the data to dense layer. I also used RELU activation for 4 dense layers following with a drop out of 50% for each, and Softmax activation for the last dense layer with a unit of 11 as we have 11 classes to predict:

→ 1 x dense layer of 4096 units

→ 1 x drop out layer

→ 1 x dense layer of 1024 units

→ 1 x drop out layer

→ 1 x dense layer of 256 units

→ 1 x drop out layer

→ 1 x dense layer of 32 units

→ 1 x drop out layer

→ 1 x dense softmax layer of 11 units

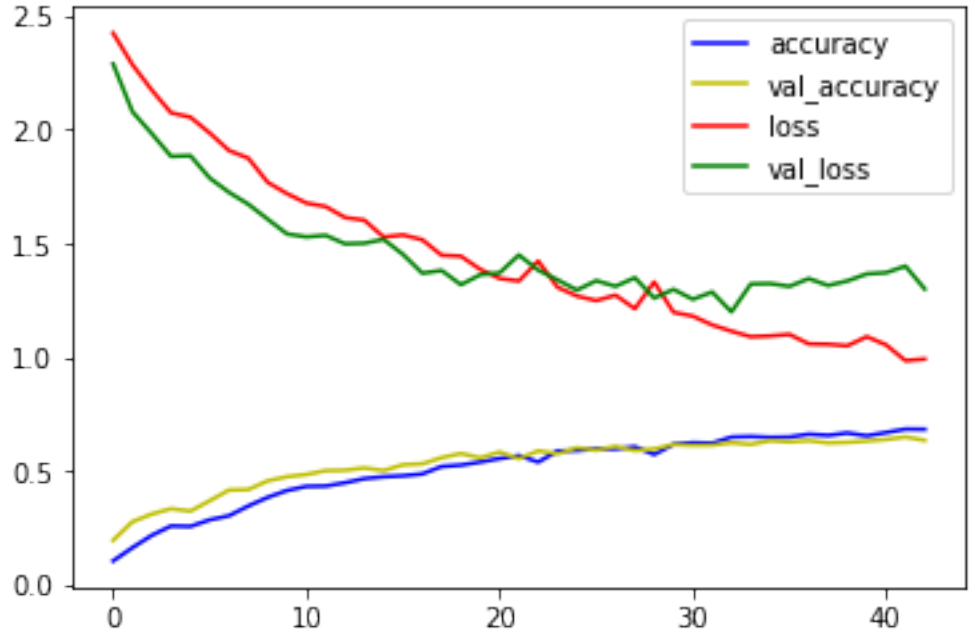
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
model (Functional)           (None, 8, 8, 512)         14714688
_________________________________________________________________
conv2d (Conv2D)              (None, 6, 6, 512)         2359808
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 4, 4, 1024)        4719616
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 2, 2, 1024)        0
_________________________________________________________________
flatten (Flatten)            (None, 4096)              0
_________________________________________________________________
dense (Dense)                (None, 4096)              16781312
_________________________________________________________________
dropout (Dropout)            (None, 4096)              0
_________________________________________________________________
dense_1 (Dense)              (None, 1024)              4195328
_________________________________________________________________
dropout_1 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_2 (Dense)              (None, 256)               262400
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_3 (Dense)              (None, 32)                8224
_________________________________________________________________
dropout_3 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_4 (Dense)              (None, 11)                363
=================================================================
Total params: 43,041,739
Trainable params: 28,327,051
Non-trainable params: 14,714,688
```

Oanh Le

CIS4930 Deep Learning Fundamentals

Project 1-2 Report

Results



Validation loss: 1.19787

Validation accuracy: 64.87%