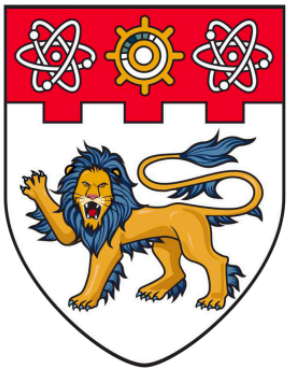


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

BS6207 ASSIGNMENT 3

LONG JINGYU

4/1/2022

QUESTION 1

MaxPool2d

For the Max pooling2D, a sample-based discretization process. The objective is to down-sample an input representation (image, hidden-layer output matrix, etc.), reducing its dimensionality and allowing for assumptions to be made about features contained in the sub-regions binned.

This is done to in part to help over-fitting by providing an abstracted form of the representation and I take the maximum of a window in the input moving with step stride.

AvgPool2d

The main mechanism of average pooling is almost the same as max pooling, the only different is that the basic operation is average.

This is done to in part to help over-fitting by providing an abstracted form of the representation, which conserve more background information, and I take the average of a window in the input moving with step stride.

Conv2d

I start with a kernel, which is simply a small matrix of weights. "kernel size" means the size of a convolutional filter, to be more specifically, it refers to the width x height of the filter mask, and for the stride, stride controls the stride for the cross-correlation, a single number or a tuple.

Conv2d with Dilation

Dilation controls the spacing between the kernel points, so just by make some modification in first Conv2d function, we can get the second Conv2d function. Dilation with different values are shown below, we are taking dilation = 2.

Conv Transpose 2d

Transposed convolution is the inverse operation of convolution, that is, the dimension of the feature is compressed, but the size is enlarged.

Flatten

To be simplified, the flatten operation is just transform our original matrix to one dimensional array.

ROI pooling

ROI, means region of interest, take the region of each channel/feature with the box specifying the coordinates (x1,y1,x2,y2). Then max pooling is applied on the ROI box with the kernel size specified with output_size parameter here.

Sigmoid

The mechanism of sigmoid function is to convert the model's output into a probability score, and its mathematical representation can be easily achieved by numpy which is 1/(1+np.exp(-x)), torch_out and my_out of Sigmoid are equal up to 4 dicimal digits.

Batch_norm

The key point of batch_norm is to calculate the mean and variance, and just following its mathematical formula, we can easily code it.

Cross Entropy Loss

Cross entropy is mainly used in classification problem and MSE is mainly used in regression problem, cross entropy loss is the log soft max here, the formula of it is that the classification results are normalized to form a probability distribution and take the log value, for MSE, it measures the average of the squares of the errors, that is, the average squared difference between the estimated values and the actual value. It is useful when training a classification problem with C classes, calculating the probability of each class for an input between 0 and 1 (sum up to 1)

MSE Loss

MAE loss, mean squared error between inputs and target.

QUESTION 2A

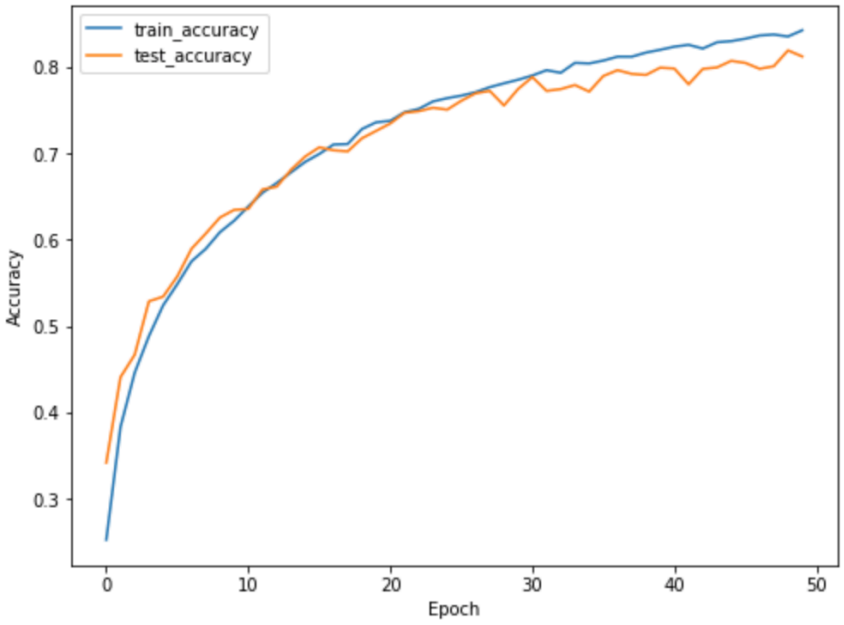
```
Using TensorFlow backend.
```

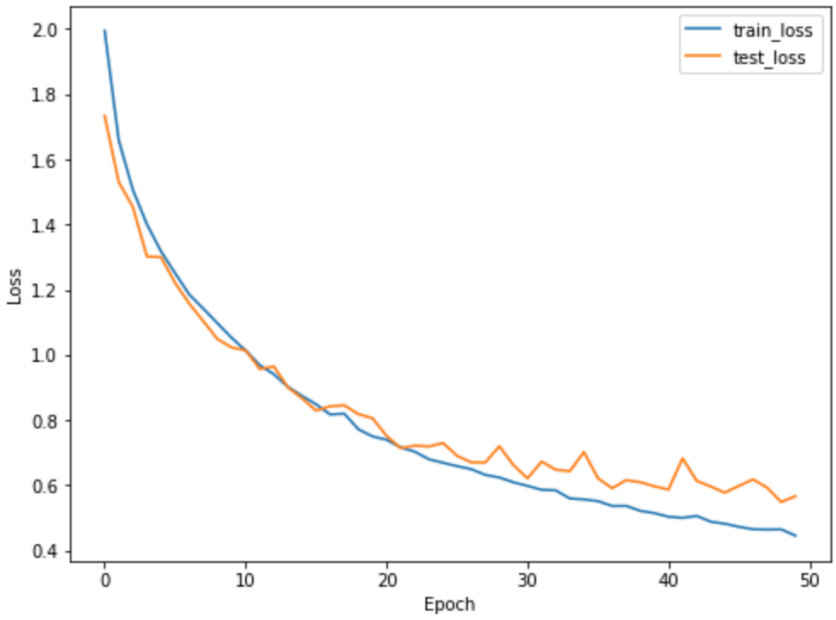
```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 32, 32, 16)        448
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 32, 32, 16)        2320
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 16, 16, 16)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 16, 16, 32)        4640
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 16, 16, 32)        9248
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 8, 8, 32)          0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 8, 8, 64)          18496
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 8, 8, 64)          36928
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 4, 4, 64)          0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 4, 4, 128)         73856
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 4, 4, 128)         147584
_________________________________________________________________
average_pooling2d_1 (Average (None, 1, 1, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                1290
=================================================================
Total params: 294,810
Trainable params: 294,810
Non-trainable params: 0
_________________________________________________________________
```

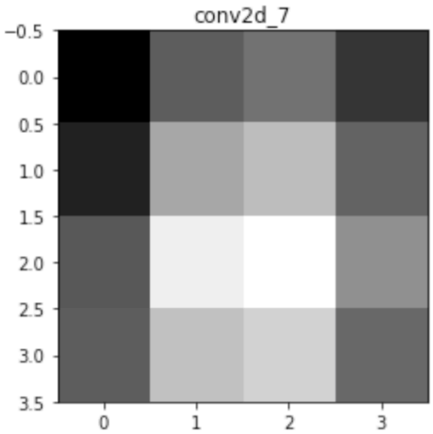
```
Epoch 00050: val_accuracy did not improve from 0.81930
```

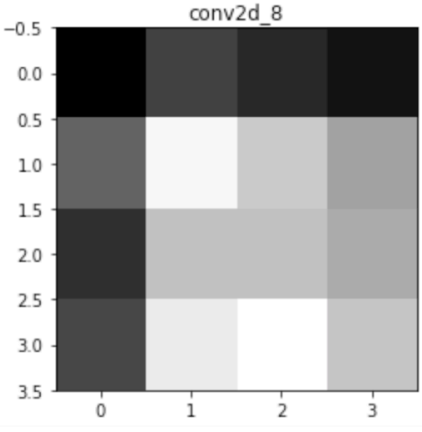
```
model_filepath = os.path.join(save_dir,'epoch49_acc0.82.hdf5')
model1.load_weights(model_filepath)
acc = model1.evaluate(x_test, y_test, verbose=1)[1]
print('Test Accuracy:', acc)
```

```
10000/10000 [==============================] - 6s 624us/step
Test Accuracy: 0.8192999958992004
```


train_accuracy
test_accuracy
Accuracy
Epoch


train_loss
test_loss
Loss
Epoch


conv2d_7


conv2d_8

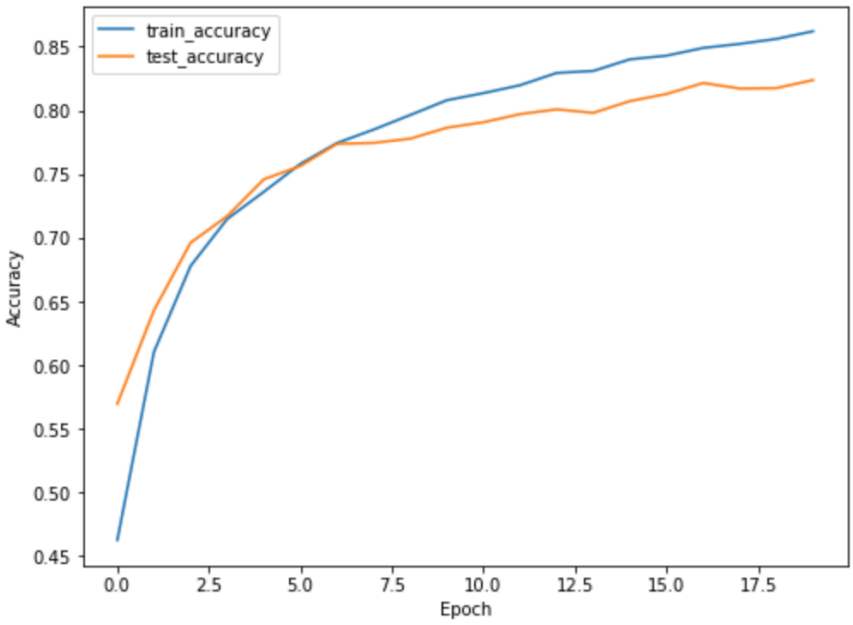
QUESTION 2B

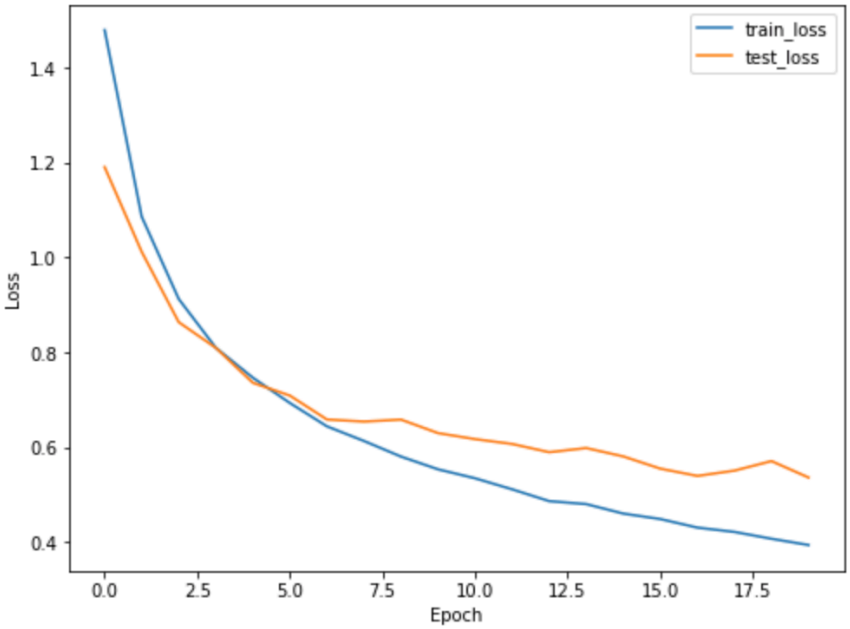
```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_9 (Conv2D)            (None, 32, 32, 16)        448
_________________________________________________________________
batch_normalization_1 (Batch (None, 32, 32, 16)        64
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 32, 32, 16)        2320
_________________________________________________________________
batch_normalization_2 (Batch (None, 32, 32, 16)        64
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 16, 16, 16)        0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 16, 16, 32)        4640
_________________________________________________________________
batch_normalization_3 (Batch (None, 16, 16, 32)        128
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 16, 16, 32)        9248
_________________________________________________________________
batch_normalization_4 (Batch (None, 16, 16, 32)        128
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 8, 8, 32)          0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 8, 8, 64)          18496
_________________________________________________________________
batch_normalization_5 (Batch (None, 8, 8, 64)          256
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 8, 8, 64)          36928
_________________________________________________________________
batch_normalization_6 (Batch (None, 8, 8, 64)          256
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 4, 4, 64)          0
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 4, 4, 128)         73856
_________________________________________________________________
batch_normalization_7 (Batch (None, 4, 4, 128)         512
_________________________________________________________________
conv2d_16 (Conv2D)           (None, 4, 4, 128)         147584
_________________________________________________________________
batch_normalization_8 (Batch (None, 4, 4, 128)         512
_________________________________________________________________
average_pooling2d_2 (Average (None, 1, 1, 128)         0
_________________________________________________________________
flatten_2 (Flatten)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                1290
=================================================================
Total params: 296,730
Trainable params: 295,770
Non-trainable params: 960
_________________________________________________________________
```

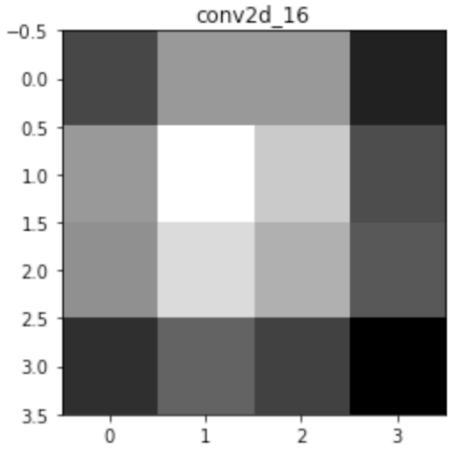
Epoch 00020: val_accuracy improved from 0.82150 to 0.82380, saving model to ./model2\epoch20_acc0.82.hdf5

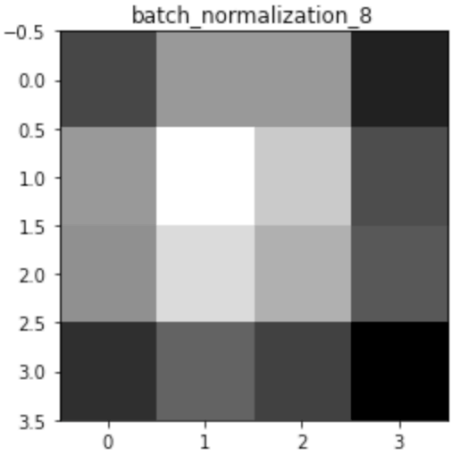
```
model_filepath = os.path.join(save_dir,'epoch20_acc0.82.hdf5')
model2.load_weights(model_filepath)
acc = model2.evaluate(x_test, y_test, verbose=1)[1]
print('Test Accuracy:', acc)
```

```
10000/10000 [==============================] - 7s 712us/step
Test Accuracy: 0.8238000273704529
```


train_accuracy
test_accuracy
Accuracy
Epoch


train_loss
test_loss
Loss
Epoch


conv2d_16


batch_normalization_8

A Test Accuracy is 0.819, and B is 0.8238, so the result of B present better due to the bn. Batch norm reduces the gradient dispersion, improves the training speed, greatly speeds up the convergence process, and also increases the classification effect.The callback process is much simpler, the initialization requirements are not so high, and you can use large learning rates. However, batch normalization relies on the size of the batch, and when the batch value is small, the calculated mean and variance are unstable.