Report

Wednesday, October 10, 2018 1:19 PM

- ## You are expected to provide a recommendation for the best model you would recommend for classification. Which model (with parameter values) would you choose and why?

  This is the model(with parameter values) I choose:

```
batch_size = 128
num_classes = 10
epochs = 100
learning_rate = 0.01
dropout_rate = 0.4


model = Sequential()
model.add(Dense(1024, activation='relu',input_shape=(3072,)))
model.add(Dropout(dropout_rate))
model.add(Dense(1024, activation='relu'))
model.add(Dropout(dropout_rate))
model.add(Dense(512, activation='relu'))
model.add(Dropout(dropout_rate))
model.add(Dense(10, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))

model.summary()

model.compile(loss='categorical_crossentropy',
              optimizer=SGD(momentum=0.9),
              metrics=['accuracy'])
```

  I choose this model is because I've done many experiments with different combinations of MLP architectures, parameter values, and optimizers, and this model is the best among all of my experiments. Among all my experiments, I found mlp seems to only be able to acquire aorund 50% acc, which means, for cifar10 data, if higher acc needed, better neural networks models like CNN need to be used. In this case, my strategy will be using the model as simple as possible while it can aslo achieve the same 50%-ish acc.

  I choose two expeiments from all those experiments as EX1 and EX2 to compare.
  EX1 has 6 layers with more trainable weights than the best model. We can see from the training chart, the training acc is more than 10% larger than validation acc. This means the model had met an overfitting situation. So in EX2, I was trying to improve the ability of the model by add more weights meanwhile add regularizers in each of the dense layer to improve its capability of generalization.
  Ex2, there are more neurals in each layer, which means the model should have better capability to learn from the training data. Meanwhile, with the kernel_regularizer added, the model should not easily overfitted. However, we can see from the training chart, the training acc was always around 33% meanwhile the training loss was hardly reduced during 100 epochs, which means this model is hardly to learn from the training data and it performed badly on validation data. So I guess the cause might be those kernel_regularizers, it weakens the model too much.

- ## Comment on how good your model is ?

  To see how good the model is, we need to use an appropriate metirc. Since our task is a classification, we use 'categorical_crossentropy' here. The perfromance of my model on testing data after 100 epoch training is:
  Test loss: 1.295480798149109 Test accuracy: 0.5424

- ## Does it overfit/underfit data ?

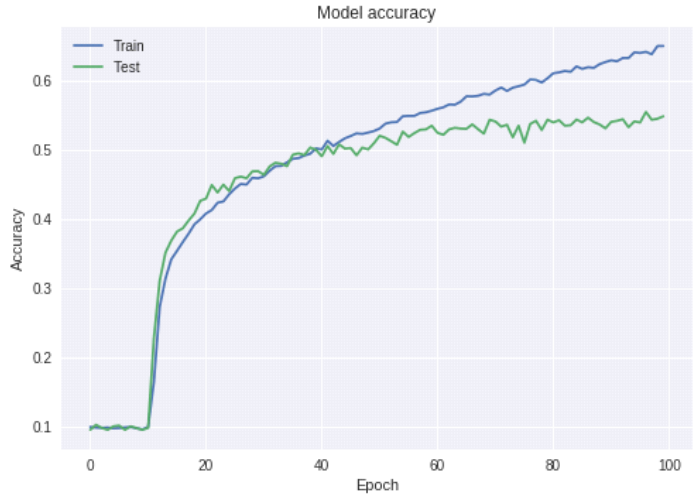
  The left below is the first time I ran my model with a dropout rate at 0.3. We can see from the acc. char that

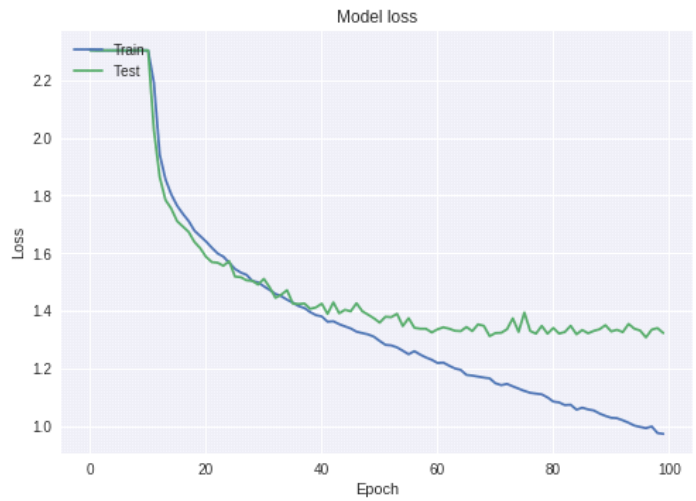
the model had been overfitted after 40 epochs. The training acc. went higher and higher although the validation acc. growing very slow.

- ## What could you do to improve the model?

  When the model overfitted, it is saying that our model lacks the ability of generalization. At this time, what we can do is either adding regularizers on kernels and biases or using a larger dropout rate. Because of the failure of using regularizers before, I decided to use a larger dropout rate, thus I turned it to 0.4. As you can see in the right below, by raising 10% of dropout rate, my model overfitted at 45 epochs! The point of overfitting had been postponed 5 epochs.

batch_size = 128
num_classes = 10
epochs = 100
learning_rate = 0.01
dropout_rate = 0.3



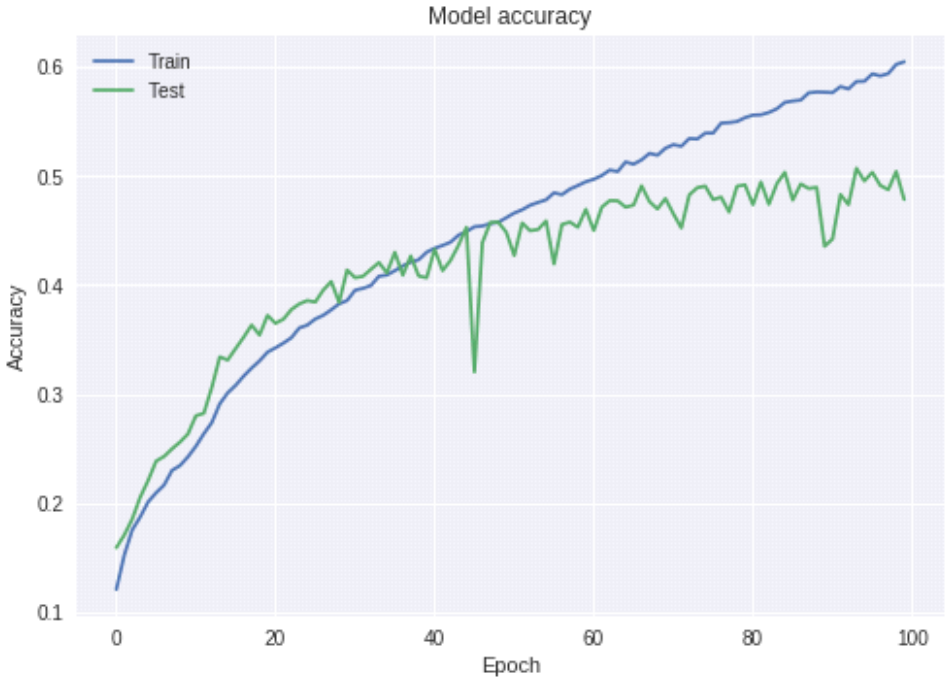


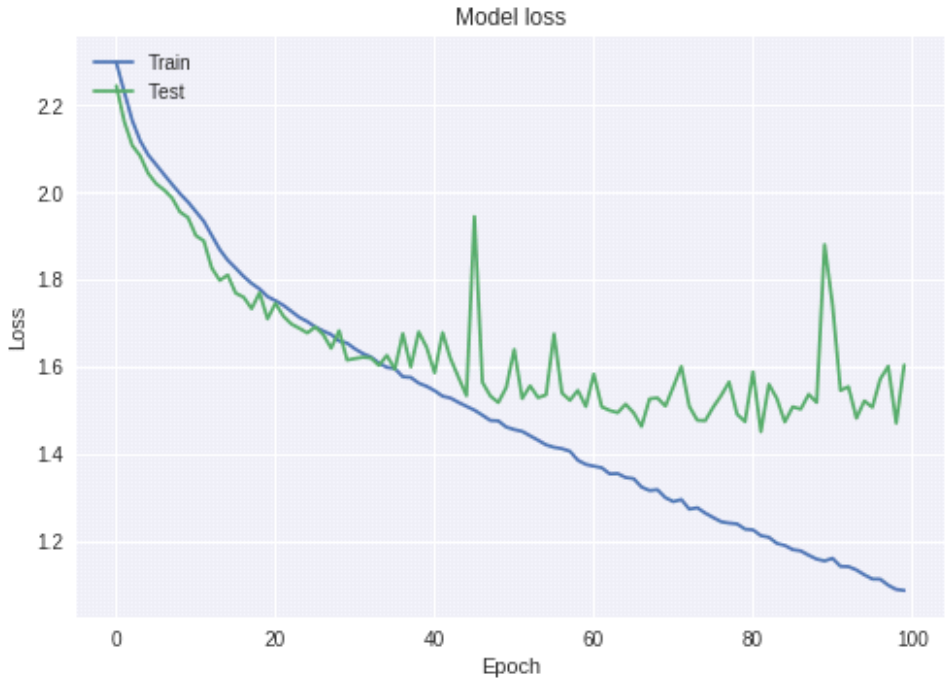
batch_size = 128
num_classes = 10
epochs = 100
learning_rate = 0.01
dropout_rate = 0.4

Ex1

Thursday, October 11, 2018 9:55 AM

```
batch_size = 128
num_classes = 10
epochs = 100
learning_rate = 0.01
dropout_rate = 0.3



model = Sequential()
model.add(Dense(1024, activation='relu',input_shape=(3072,)))
model.add(Dropout(0.1))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))


model.summary()


model.compile(loss='categorical_crossentropy',
        optimizer=SGD(momentum=0.9),
        metrics=['accuracy'])
```


Model accuracy
Train
Test
Accuracy
Epoch
0.1
0.2
0.3
0.4
0.5
0.6
0
20
40
60
80
100

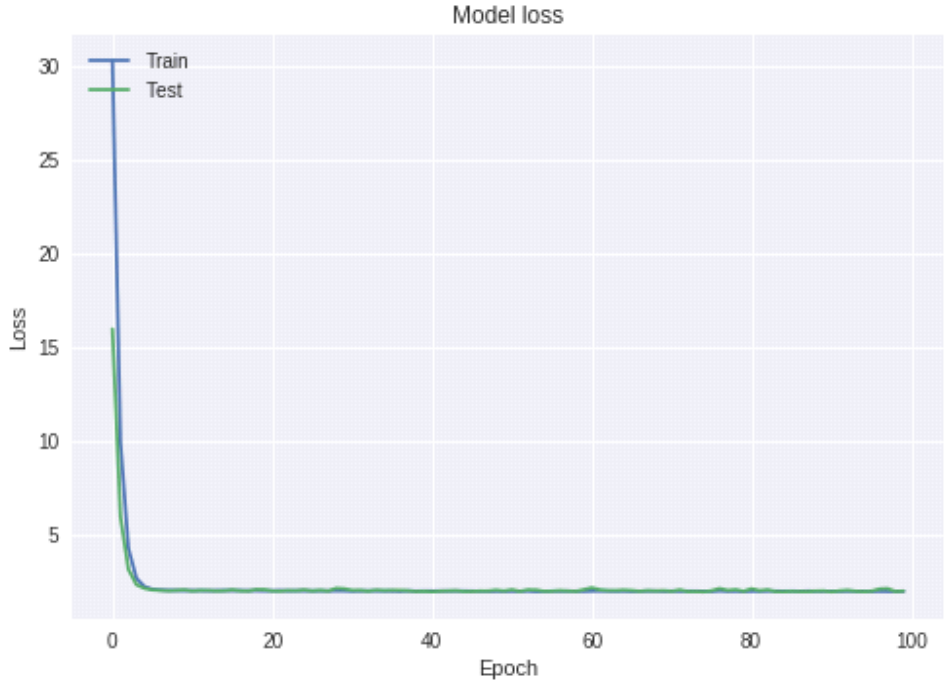

Model loss
Train
Test
Loss
Epoch
12
14
16
18
20
22
0
20
40
60
80
100

Ex2

Thursday, October 11, 2018 9:55 AM

```
batch_size = 128
num_classes = 10
epochs = 100
learning_rate = 0.01
dropout_rate = 0.3



model = Sequential()
model.add(Dense(1024, activation='relu',kernel_regularizer=regularizers.l2(0.01),input_shape=(3072,)))
#model.add(Dropout(dropout_rate))
model.add(Dense(1024, activation='relu',kernel_regularizer=regularizers.l2(0.01)))
#model.add(Dropout(dropout_rate))
model.add(Dense(1024, activation='relu',kernel_regularizer=regularizers.l2(0.01)))
#model.add(Dropout(dropout_rate))
model.add(Dense(512, activation='relu',kernel_regularizer=regularizers.l2(0.01)))
#model.add(Dropout(dropout_rate))
model.add(Dense(512, activation='relu',kernel_regularizer=regularizers.l2(0.01)))
#model.add(Dropout(dropout_rate))
model.add(Dense(10, activation='relu',kernel_regularizer=regularizers.l2(0.01)))
model.add(Dense(num_classes, activation='softmax'))


model.summary()


model.compile(loss='categorical_crossentropy',
        optimizer=SGD(momentum=0.9),
        metrics=['accuracy'])
```


Model accuracy
Train
Test
Accuracy
0.36
0.34
0.32
0.30
0.28
0.26
0.24
0
20
40
60
80
100
Epoch


Model loss
Train
Test
Loss
30
25
20
15
10
5
0
20
40
60
80
100
Epoch