Longfei Zeng
Student ID:0699514
My last three digit is 514, therefore my Database is Caltech252, backbone is VGG-16 and shallow classifiers is SVM. But I want to challenge the Deep NN, like DenseNet121.

1. Refer DenseNet121 from Keras.application:

```
if resume == False:
    basemodel = DenseNet121(include_top=False, weights=None, input_shape=(32, 32, 3), pooling='avg')
else:
    basemodel = DenseNet121(include_top=False, weights='imagenet', input_shape=(32, 32, 3), pooling='avg')
```

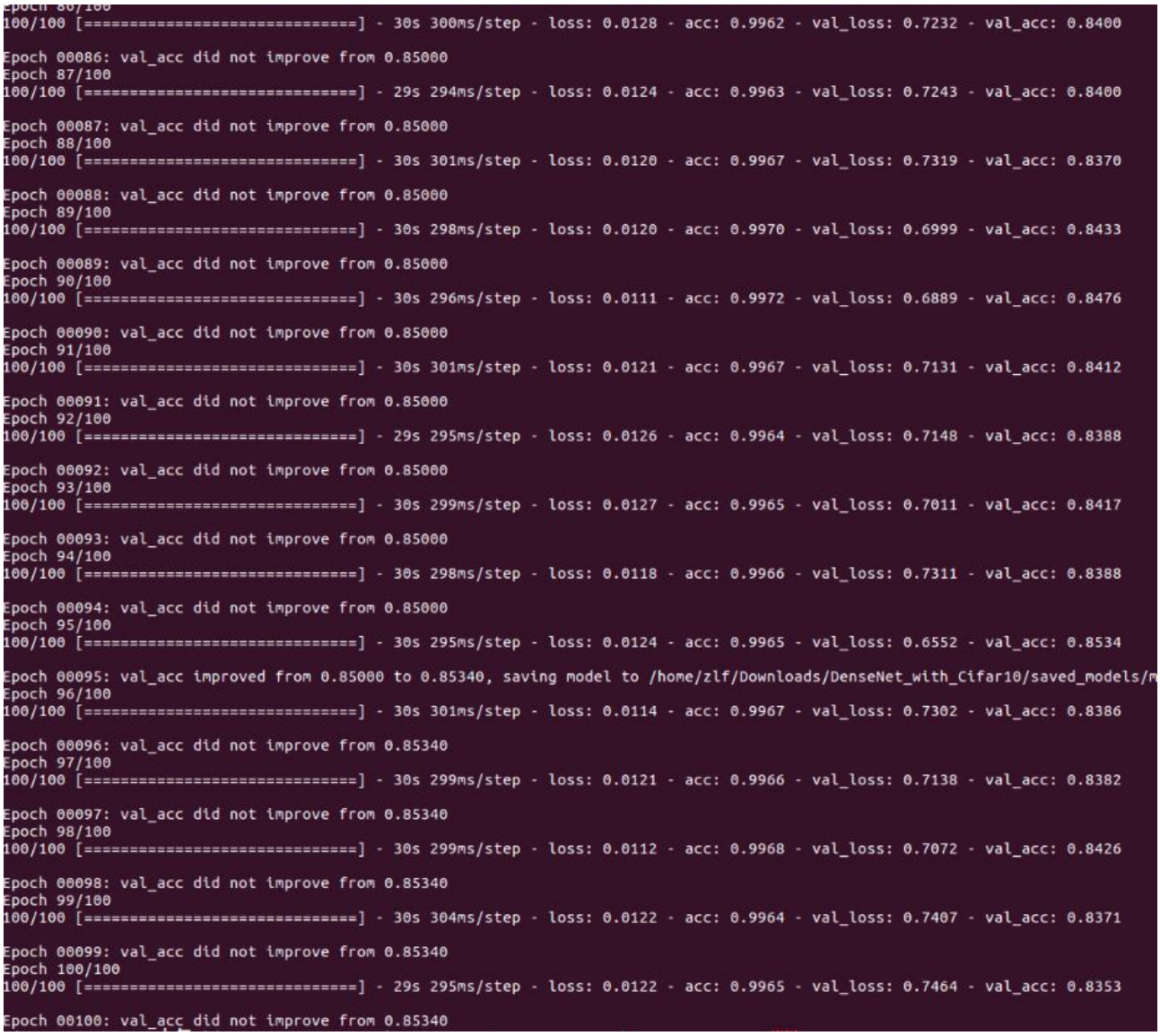
Use pretrained weights by “imagenet”, the other is None.
Set head Dense Layers asbelow:

```
x = basemodel.output
x = Dense(100, activation='sigmoid', name='fc100')(x)
prediction = Dense(nb_classes, activation='softmax', name='fc10')(x)
model = Model(input=basemodel.input, output=prediction)
```

The DenseNet121 have 121 Conv2D layers, frozen all of them is unreasonable, because Imagenet and Caltech252 are totally different from Classes Number and Structure like size and variety. For pretrained network, after my practice, frozen the tail 16 layers, they are basic features layer, it can save time to extract low-level features like texture and direction features. And model give wide space for high layers to learn more and more abstract features for Caltech252.
For None pretrained network, every layer can be trained.

```
if resume == False:
    for layer in basemodel.layers:
        layer.trainable = True
else:
    print("Setting Trainable Permission")
    for layer in basemodel.layers:
        layer.trainable = True
    basemodel.layers[1].trainable = False
    basemodel.layers[2].trainable = False
    basemodel.layers[3].trainable = False
    basemodel.layers[4].trainable = False
    basemodel.layers[5].trainable = False
    basemodel.layers[6].trainable = False
    """ for i in range(16):
        basemodel.layers[(i+1)].trainable = False"""
```

After run Pretrained VGG-16 3 tiems, I get 3 accuracys:



The average accuracy is 85.53%.
PS:
There is a test result if I frozen all DenseNet121 layers and only train last Dense and Softmax layers, the model performed very bad, like 10% or 20% accuracy the reason I summary as the Dataset is totally different, nearly no common attributes. And for DenseNet121 is very deep model, and Relay on Concatenate layers to merge low and high level features, frozen all layers is crazy to reduce the generalization ability of model.

```
Epoch 1/100
100/100 [==============================] - 27s 274ms/step - loss: 1.8480 - acc: 0.3711 - val_loss: 3.6045 - val_acc: 0.0960

Epoch 00001: val_acc improved from -inf to 0.09600, saving model to /home/zlf/Downloads/DenseNet_with_Cifar10/saved_models/model_
Epoch 2/100
100/100 [==============================] - 21s 211ms/step - loss: 1.4778 - acc: 0.4926 - val_loss: 4.1658 - val_acc: 0.0986

Epoch 00002: val_acc improved from 0.09600 to 0.09860, saving model to /home/zlf/Downloads/DenseNet_with_Cifar10/saved_models/mod
Epoch 3/100
100/100 [==============================] - 21s 212ms/step - loss: 1.3985 - acc: 0.5159 - val_loss: 4.6734 - val_acc: 0.1017

Epoch 00003: val_acc improved from 0.09860 to 0.10170, saving model to /home/zlf/Downloads/DenseNet_with_Cifar10/saved_models/mod
Epoch 4/100
100/100 [==============================] - 22s 219ms/step - loss: 1.3529 - acc: 0.5280 - val_loss: 4.7885 - val_acc: 0.1023

Epoch 00004: val_acc improved from 0.10170 to 0.10230, saving model to /home/zlf/Downloads/DenseNet_with_Cifar10/saved_models/mod
Epoch 5/100
100/100 [==============================] - 21s 215ms/step - loss: 1.3293 - acc: 0.5334 - val_loss: 5.0537 - val_acc: 0.0959

Epoch 00005: val_acc did not improve from 0.10230
Epoch 6/100
100/100 [==============================] - 22s 217ms/step - loss: 1.3152 - acc: 0.5408 - val_loss: 5.1930 - val_acc: 0.1001

Epoch 00006: val_acc did not improve from 0.10230
Epoch 7/100
100/100 [==============================] - 21s 211ms/step - loss: 1.2946 - acc: 0.5462 - val_loss: 5.1892 - val_acc: 0.0963

Epoch 00007: val_acc did not improve from 0.10230
Epoch 8/100
100/100 [==============================] - 22s 215ms/step - loss: 1.2825 - acc: 0.5527 - val_loss: 5.3602 - val_acc: 0.0973

Epoch 00008: val_acc did not improve from 0.10230
Epoch 9/100
100/100 [==============================] - 21s 213ms/step - loss: 1.2764 - acc: 0.5525 - val_loss: 5.3101 - val_acc: 0.1025

Epoch 00009: val_acc improved from 0.10230 to 0.10250, saving model to /home/zlf/Downloads/DenseNet_with_Cifar10/saved_models/mod
Epoch 10/100
100/100 [==============================] - 22s 222ms/step - loss: 1.2602 - acc: 0.5595 - val_loss: 5.4127 - val_acc: 0.1016

Epoch 00010: val_acc did not improve from 0.10250
Epoch 11/100
100/100 [==============================] - 21s 213ms/step - loss: 1.2522 - acc: 0.5590 - val_loss: 5.4030 - val_acc: 0.1005

Epoch 00011: val_acc did not improve from 0.10250
Epoch 12/100
100/100 [==============================] - 22s 216ms/step - loss: 1.2394 - acc: 0.5648 - val_loss: 5.4538 - val_acc: 0.0973

Epoch 00012: val_acc did not improve from 0.10250
Epoch 13/100
100/100 [==============================] - 21s 214ms/step - loss: 1.2352 - acc: 0.5673 - val_loss: 5.4356 - val_acc: 0.1003

Epoch 00013: val_acc did not improve from 0.10250
Epoch 14/100
100/100 [==============================] - 21s 211ms/step - loss: 1.2201 - acc: 0.5713 - val_loss: 5.3914 - val_acc: 0.1037
```

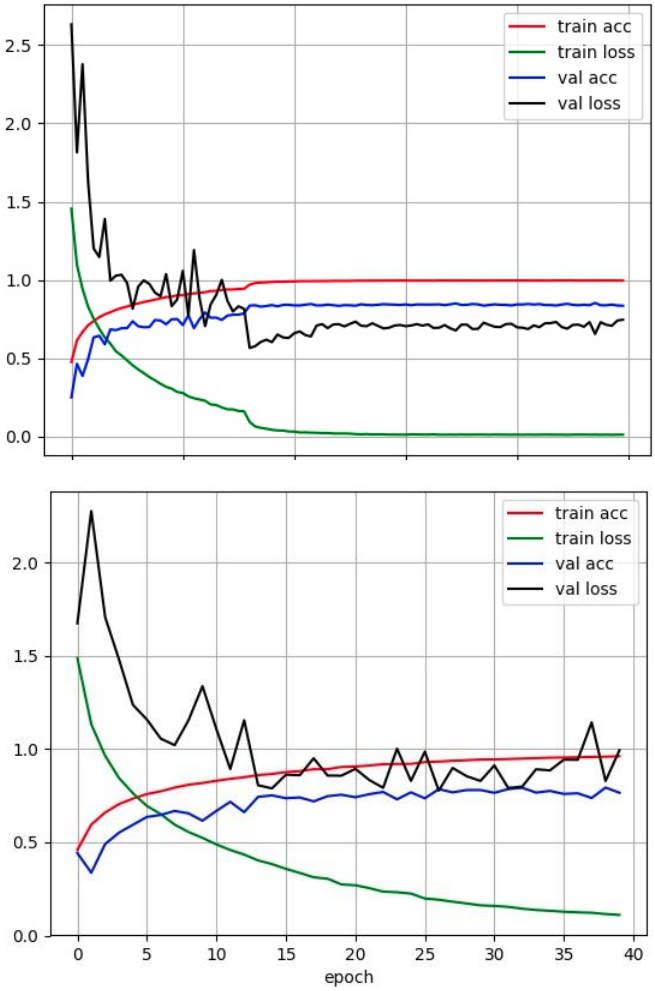
After run normal VGG-16 3 tiems, I get 3 accuracys:

```
Epoch 00092: val_acc did not improve from 0.85000
Epoch 93/100
100/100 [==============================] - 30s 299ms/step - loss: 0.0127 - acc: 0.9965 - val_loss: 0.7011 - val_acc: 0.8417

Epoch 00093: val_acc did not improve from 0.85000
Epoch 94/100
100/100 [==============================] - 30s 298ms/step - loss: 0.0118 - acc: 0.9966 - val_loss: 0.7311 - val_acc: 0.8388

Epoch 00094: val_acc did not improve from 0.85000
Epoch 95/100
100/100 [==============================] - 30s 295ms/step - loss: 0.0124 - acc: 0.9965 - val_loss: 0.6552 - val_acc: 0.8534

Epoch 00095: val_acc improved from 0.85000 to 0.85340, saving model to /home/zlf/Downloads/DenseNet_with_Cifar10/saved_models/model_95-0.85.hdf5
Epoch 96/100
100/100 [==============================] - 30s 301ms/step - loss: 0.0114 - acc: 0.9967 - val_loss: 0.7302 - val_acc: 0.8386

Epoch 00096: val_acc did not improve from 0.85340
Epoch 97/100
100/100 [==============================] - 30s 299ms/step - loss: 0.0121 - acc: 0.9966 - val_loss: 0.7138 - val_acc: 0.8382

Epoch 00097: val_acc did not improve from 0.85340
Epoch 98/100
100/100 [==============================] - 30s 299ms/step - loss: 0.0112 - acc: 0.9968 - val_loss: 0.7072 - val_acc: 0.8426

Epoch 00098: val_acc did not improve from 0.85340
Epoch 99/100
100/100 [==============================] - 30s 304ms/step - loss: 0.0122 - acc: 0.9964 - val_loss: 0.7407 - val_acc: 0.8371

Epoch 00099: val_acc did not improve from 0.85340
Epoch 100/100
100/100 [==============================] - 29s 295ms/step - loss: 0.0122 - acc: 0.9965 - val_loss: 0.7464 - val_acc: 0.8353

Epoch 00100: val_acc did not improve from 0.85340
#( 02/22/19@10:34上午 )( zlf@zlf-Oryx-Pro ):~/Downloads/DenseNet_with_Cifar10@master✗✗✗
```

The average accuracy is also converged around 85.43%.

Let us compare the training processing:

the picture showes the best model training and testing processing below,top is Pretrained weights network, below is normal DenseNet121.



We can see that Pretrained network converged faster than normal one, because it have already extracted some good performance low-level and some high-level features, it help network growing faster, but the key is find a frozen boundary help model to converge quickly.

Shallow network SVM.
I get best model's CNN layers output as deep feature, and then feed it into SVM classifier.

```
basemodel = DenseNet121(include_top=False, weights=None, input_shape=(32, 32, 3), pooling='avg')
deep_feature = basemodel.output
x = Dense(100, activation='sigmoid', name='fc100')(deep_feature)
prediction = Dense(10, activation='softmax', name='fc10')(x)
model = Model(input=basemodel.input, output=prediction)

model.load_weights(check_point_file)
submodel = Model(input=basemodel.input, output=deep_feature)

(x_train, y_train), (x_test, y_test) = cifar10.load_data()

deep_feature1 = submodel.predict(x_train)
deep_feature2 = submodel.predict(x_test)

start = time.clock()

clf = SVC(kernel='poly')
clf.fit(deep_feature1, y_train)
elapsed = (time.clock() - start)
print("Time used:", elapsed)

score = clf.score(deep_feature2, y_test)
print(" score: {:.6f}".format(score))
```

For basic VGG-16 it got 83.75% accuracy.

```
Time used: 537.53738336260001
 score: 0.837500
```

For pretrained VGG-16, it got 84.01% accuracy.

```
Time used: 783.38374629000031
 score:0.840100
```

So, shallow network like SVM classifier maybe have lower expression ability for deep features, and several Dense + Softmax layers can perform better than shallow one.
And properly use pretrained weights can help network to converge faster.