Ensemble NN

Version 4 of 4

Notebook
Input (1)
Output
Execution Info
Log
Comments (0)

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all file
s under the input directory
import random
import os
import cv2
from IPython.display import Image
import matplotlib.pyplot as plt
import seaborn as sns

from keras.utils import plot_model
from sklearn.metrics import classification_report
from collections import Counter
import tensorflow as tf

import keras
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ReduceLROnPlateau
from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.applications.vgg19 import VGG16

from keras import Model, layers
from keras.layers.core import Dense
from keras.optimizers import Adam, SGD
from keras.layers import GlobalMaxPooling2D, GlobalAveragePooling2D, Dropout, Dense, I
nput, Conv2D, MaxPooling2D, Flatten,MaxPooling2D
# You can write up to 20GB in the current directory (/kaggle/working/) that gets preserv
ed as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside o
f the current session
```

In [2]:

```
root_path = '/kaggle/input/intel-image-classification/'
train_pred_test_folders = os.listdir(root_path)

seg_train_folders = os.listdir(os.path.join(root_path,train_pred_test_folders[0],train
_pred_test_folders[0])) # gets seven seg_train folder names
seg_pred_folders = os.listdir(os.path.join(root_path,train_pred_test_folders[1],train_
pred_test_folders[1]))
seg_test_folders = os.listdir(os.path.join(root_path,train_pred_test_folders[2],train_
pred_test_folders[2]))
quantity_tr = {}
quantity_te = {}
for folder in seg_train_folders:
    quantity_tr[folder] = len(os.listdir(os.path.join(root_path,train_pred_test_folder
s[0],train_pred_test_folders[0],folder)))
for folder in seg_test_folders:
    quantity_te[folder] = len(os.listdir(os.path.join(root_path,train_pred_test_folder
s[2],train_pred_test_folders[2],folder)))

quantity_train = pd.DataFrame(list(quantity_tr.items()), index=range(0,len(quantity_t
r)), columns=['class', 'count'])
quantity_test = pd.DataFrame(list(quantity_te.items()), index=range(0,len(quantity_t
e)), columns=['class', 'count'])

figure, ax = plt.subplots(1,2,figsize=(18,5))
sns.barplot('class', y='count',data=quantity_train,ax=ax[0])
sns.barplot('class', y='count',data=quantity_test,ax=ax[1])

print('Number of images in the train set - ',sum(quantity_tr.values()))
print('Number of images in the test set - ',sum(quantity_te.values()))
number_of_images_in_prediction_set = len(os.listdir(os.path.join(root_path,train_pred
_test_folders[1],train_pred_test_folders[1])))
print('Number of images in prediction set - ',number_of_images_in_prediction_set)

plt.show()
```

```
Number of images in the train set -  14034
Number of images in the test set -  3000
Number of images in prediction set -  7301
```

In [3]:

```
def save_history(history, model_name):
    #convert the history.history dict to a pandas DataFrame:
    hist_df = pd.DataFrame(history.history)

    # save to json:
    hist_json_file = model_name+'_history.json'
    with open(hist_json_file, mode='w') as f:
        hist_df.to_json(f)

    # or save to csv:
    hist_csv_file = model_name+'_history.csv'
    with open(hist_csv_file, mode='w') as f:
        hist_df.to_csv(f)

def plot_accuracy_from_history(history, isincception=False):
    color = sns.color_palette()
    if(isincception == False):
        acc = history['acc']
        val_acc = history['val_acc']
    else:
        acc = history['accuracy']
        val_acc = history['val_accuracy']

    epochs = range(len(acc))

    sns.lineplot(epochs, acc, label='Training Accuracy')
    sns.lineplot(epochs, val_acc,label='Validation Accuracy')
    plt.title('Training and Validation Accuracy')
    plt.legend()
    plt.figure()
    plt.show()

def plot_loss_from_history(history):
    color = sns.color_palette()
    loss = history['loss']
    val_loss = history['val_loss']

    epochs = range(len(loss))

    sns.lineplot(epochs, loss,label='Training Loss')
    sns.lineplot(epochs, val_loss, label='Validation Loss')
    plt.title('Training and Validation Loss')
    plt.legend()
    plt.figure()
    plt.show()
```

In [4]:

```
train_DIR = '/kaggle/input/intel-image-classification/seg_train/seg_train/'
train_datagen = ImageDataGenerator(rescale = 1.0/255,shear_range=0.2,zoom_range=0.2)

# we are rescaling by 1.0/255 to normalize the rgb values if they are in range 0-255 the
values are too high for good model performance.
train_generator = train_datagen.flow_from_directory(train_DIR,
                                                    batch_size=32,
                                                    shuffle=True,
                                                    class_mode='categorical',
                                                    target_size=(150, 150))

test_DIR = '/kaggle/input/intel-image-classification/seg_test/seg_test/'
validation_datagen = ImageDataGenerator(rescale = 1.0/255) # we are only normalizing to
make the prediction. the other parameters were used for augmentation and train weights.
validation_generator = validation_datagen.flow_from_directory(test_DIR, shuffle=True,
batch_size=32, class_mode='categorical', target_size=(150, 150))
```

```
Found 14034 images belonging to 6 classes.
Found 3000 images belonging to 6 classes.
```

In [5]:

```
inv_map_classes = {v: k for k, v in validation_generator.class_indices.items()}
print(validation_generator.class_indices)
print(inv_map_classes)
```

```
{'buildings': 0, 'forest': 1, 'glacier': 2, 'mountain': 3, 'sea': 4, 'street': 5}
{0: 'buildings', 1: 'forest', 2: 'glacier', 3: 'mountain', 4: 'sea', 5: 'street'}
```

In [6]:

```
def show_few_images(number_of_examples=2, predict_using_model=None):
    figsize1, ax1 = plt.subplots(number_of_examples,len(seg_train_folders), figsize=(2
0,4*number_of_examples))
    ax1 = ax1.reshape(-1)
    ansf_fun = np.vectorize(lambda ans:ans[ans.argmax()])
    ansf_fun(ans)
    ans = 0
    for i, folder in enumerate(seg_train_folders):
        image_ids = os.listdir(os.path.join(root_path,train_pred_test_folders[0],train
_pred_test_folders[0], folder))
        for j in random.sample(range(0, len(image_ids)) for i in range(0,number_of_examp
les)):
            img = image_ids[j]
            img_path = os.path.join(root_path,train_pred_test_folders[0],train_pred_te
st_folders[0], folder,img)
            display = cv2.imread(img_path)
            plt.axis('off')
            ax1[ans].imshow(display)
            title = 'True:'+folder
            if(predict_using_model:
                predicted_classnum = inv_map_classes[np.argmax(inception_best_model.p
redict(tf.nn.array(display)))]
                title = title+'\nPredict :'+predicted_classnum
            ax1[ans].set_title(title)
            ans+=1
```

In [7]:

```
show_few_images()
```

In [8]:

```
tf.keras.backend.clear_session()
```

In [9]:

```
#random architecture
benchmark_model = Sequential()
# We have to define input_shape(height, height, channels) - we have already created
# the images to be of size height size 32
# 32 because our model takes 32x32 with 3 color channels will be input into this layer
benchmark_model.add(Conv2D(128, kernel_size=7, activation='relu', input_shape=(150,15
0,3)))
benchmark_model.add(MaxPooling2D(pool_size=(4,4), strides=(2,2)))
benchmark_model.add(Conv2D(64, kernel_size=5, activation='relu'))
benchmark_model.add(MaxPooling2D(pool_size=(4,4), strides=(2,2)))
benchmark_model.add(Flatten())
benchmark_model.add(Dense(128,activation='relu'))
benchmark_model.add(Dense(6,activation='softmax'))

benchmark_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['a
cc'])

benchmark_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 144, 144, 128)     18944
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 71, 71, 128)       0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 67, 67, 64)        204864
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 32, 32, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 65536)             0
_________________________________________________________________
dense (Dense)                (None, 128)               8388736
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 774
=================================================================
Total params: 8,613,318
Trainable params: 8,613,318
Non-trainable params: 0
_________________________________________________________________
```

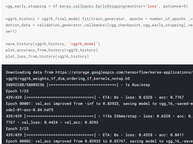
In [10]:

```
# plot_model(benchmark_model, to_file='model.png',show_shapes=True, show_layer_names=Tru
e)
# from IPython.display import Image,clear_output
# Image('model.png')#,clear_output(wait=True)
```

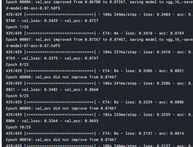
In [11]:

```
filepath = "benchmark_model-{epoch:02d}-{val_acc:.2f}.hdf5"
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.05, patience=2, min_lr=0.00
0001)
checkpoint = tf.keras.callbacks.ModelCheckpoint(filepath, monitor='val_acc', verbose=1,
save_best_only=True, mode='max')
# Keep only a single checkpoint, the best over test accuracy.
history = benchmark_model.fit(train_generator,epochs=20, verbose=1, validation_data =
validation_generator,callbacks=[reduce_lr,early_stopping,checkpoint])

save_history(history, 'benchmark_model')
benchmark_model.save("filename")
plot_accuracy_from_history(history)
plot_loss_from_history(history)
```

```
Epoch 1/20
439/439 [==============================] - ETA: 0s - loss: 1.1835 - acc: 0.5322
Epoch 00001: val_acc improved from inf to 0.47111, saving model to benchmark_mod
el-01-0.47.hdf5
439/439 [==============================] - 90s 205ms/step - loss: 1.1835 - acc: 0.
5322 - val_loss: 0.9961 - val_acc: 0.4713
Epoch 2/20
439/439 [==============================] - ETA: 0s - loss: 0.9777 - acc: 0.6006
Epoch 00002: val_acc improved from 0.47133 to 0.62633, saving model to benchmark_
model-02-0.63.hdf5
439/439 [==============================] - 100s 240ms/step - loss: 0.9777 - acc: 0.
6006 - val_loss: 0.9204 - val_acc: 0.6263
Epoch 3/20
439/439 [==============================] - ETA: 0s - loss: 0.8681 - acc: 0.6678
Epoch 00003: val_acc improved from 0.62633 to 0.70667, saving model to benchmark_
model-03-0.71.hdf5
439/439 [==============================] - 107s 244ms/step - loss: 0.8681 - acc: 0.
6678 - val_loss: 0.7750 - val_acc: 0.7067
Epoch 4/20
439/439 [==============================] - ETA: 0s - loss: 0.8098 - acc: 0.7002
Epoch 00004: val_acc did not improve from 0.70667
439/439 [==============================] - 105s 240ms/step - loss: 0.8098 - acc: 0.
7002 - val_loss: 0.7951 - val_acc: 0.7042
Epoch 5/20
439/439 [==============================] - ETA: 0s - loss: 0.7226 - acc: 0.7355
Epoch 00005: val_acc improved from 0.70667 to 0.72800, saving model to benchmark_
model-05-0.73.hdf5
439/439 [==============================] - 105s 240ms/step - loss: 0.7226 - acc: 0.
7355 - val_loss: 0.7207 - val_acc: 0.7280
Epoch 6/20
439/439 [==============================] - ETA: 0s - loss: 0.6779 - acc: 0.7526
Epoch 00006: val_acc improved from 0.72800 to 0.75033, saving model to benchmark_
model-06-0.75.hdf5
439/439 [==============================] - 104s 237ms/step - loss: 0.6779 - acc: 0.
7526 - val_loss: 0.6735 - val_acc: 0.7503
Epoch 7/20
439/439 [==============================] - ETA: 0s - loss: 0.6435 - acc: 0.7672
Epoch 00007: val_acc improved from 0.75033 to 0.76033, saving model to benchmark_
model-07-0.77.hdf5
439/439 [==============================] - 105s 241ms/step - loss: 0.6435 - acc: 0.
7672 - val_loss: 0.6324 - val_acc: 0.7603
Epoch 8/20
439/439 [==============================] - ETA: 0s - loss: 0.6165 - acc: 0.7723
Epoch 00008: val_acc did not improve from 0.76033
439/439 [==============================] - 99s 226ms/step - loss: 0.6165 - acc: 0.7
723 - val_loss: 0.6768 - val_acc: 0.7570
Epoch 9/20
439/439 [==============================] - ETA: 0s - loss: 0.5576 - acc: 0.7939
Epoch 00009: val_acc did not improve from 0.76033
439/439 [==============================] - 98s 223ms/step - loss: 0.5576 - acc: 0.7
939 - val_loss: 0.6450 - val_acc: 0.7647
Epoch 10/20
439/439 [==============================] - ETA: 0s - loss: 0.5373 - acc: 0.8042
Epoch 00010: val_acc did not improve from 0.76033
439/439 [==============================] - 98s 223ms/step - loss: 0.5373 - acc: 0.8
042 - val_loss: 0.6808 - val_acc: 0.7623
Epoch 11/20
439/439 [==============================] - ETA: 0s - loss: 0.4999 - acc: 0.8229
Epoch 00011: val_acc improved from 0.76033 to 0.77200, saving model to benchmark_
model-11-0.77.hdf5
439/439 [==============================] - 99s 226ms/step - loss: 0.4999 - acc: 0.8
229 - val_loss: 0.6401 - val_acc: 0.7720
Epoch 12/20
439/439 [==============================] - ETA: 0s - loss: 0.4557 - acc: 0.8372
Epoch 00012: val_acc did not improve from 0.77200
439/439 [==============================] - 100s 227ms/step - loss: 0.4557 - acc: 0.
8372 - val_loss: 0.6942 - val_acc: 0.7723
Epoch 13/20
439/439 [==============================] - ETA: 0s - loss: 0.3615 - acc: 0.8765
Epoch 00013: val_acc improved from 0.77200 to 0.79033, saving model to benchmark_
model-13-0.80.hdf5
439/439 [==============================] - 104s 236ms/step - loss: 0.3615 - acc: 0.
8765 - val_loss: 0.6179 - val_acc: 0.7903
Epoch 14/20
439/439 [==============================] - ETA: 0s - loss: 0.3427 - acc: 0.8754
Epoch 00014: val_acc improved from 0.79033 to 0.79633, saving model to benchmark_
model-14-0.80.hdf5
439/439 [==============================] - 105s 240ms/step - loss: 0.3427 - acc: 0.
8754 - val_loss: 0.6204 - val_acc: 0.7963
Epoch 15/20
439/439 [==============================] - ETA: 0s - loss: 0.3240 - acc: 0.8849
Epoch 00015: val_acc improved from 0.79633 to 0.80000, saving model to benchmark_
model-15-0.80.hdf5
439/439 [==============================] - 110s 251ms/step - loss: 0.3240 - acc: 0.
8849 - val_loss: 0.6329 - val_acc: 0.8000
Epoch 16/20
439/439 [==============================] - ETA: 0s - loss: 0.3222 - acc: 0.8880
Epoch 00016: val_acc did not improve from 0.80000
439/439 [==============================] - 112s 248ms/step - loss: 0.3222 - acc: 0.
8880 - val_loss: 0.6387 - val_acc: 0.7967
Epoch 17/20
439/439 [==============================] - ETA: 0s - loss: 0.3185 - acc: 0.8886
Epoch 00017: val_acc did not improve from 0.80000
439/439 [==============================] - 112s 248ms/step - loss: 0.3185 - acc: 0.
8886 - val_loss: 0.6387 - val_acc: 0.7967
Epoch 18/20
439/439 [==============================] - ETA: 0s - loss: 0.3068 - acc: 0.8930
Epoch 00018: val_acc improved from 0.80000 to 0.80033, saving model to benchmark_
model-18-0.80.hdf5
439/439 [==============================] - 114s 260ms/step - loss: 0.3068 - acc: 0.
8930 - val_loss: 0.6329 - val_acc: 0.8003
Epoch 19/20
439/439 [==============================] - ETA: 0s - loss: 0.2895 - acc: 0.8925
Epoch 00019: val_acc did not improve from 0.80033
439/439 [==============================] - 114s 260ms/step - loss: 0.2895 - acc: 0.
8925 - val_loss: 0.6389 - val_acc: 0.7967
Epoch 20/20
439/439 [==============================] - ETA: 0s - loss: 0.2546 - acc: 0.8976
Epoch 00020: val_acc did not improve from 0.80033
439/439 [==============================] - 114s 260ms/step - loss: 0.2546 - acc: 0.
8976 - val_loss: 0.6337 - val_acc: 0.7993
```




```
<Figure size 432x288 with 0 Axes>
```




```
<Figure size 432x288 with 0 Axes>
```

In [50]:

```
filepath = "vgg16_saved-model-{epoch:02d}-{val_acc:.2f}.hdf5"
vgg16_model = ModelCheckpoint(filepath, monitor='val_acc', include_top=False, input_shape=
(150,150,3))
for layers in vgg16_model.layers:
    layers.trainable=False
last_output = vgg16_model.layers[-1].output
vgg_x = Flatten()(last_output)
vgg_x = Dense(512, activation = 'relu')(vgg_x)
vgg_x = Dense(3, activation = 'softmax')(vgg_x)

vgg16_final_model = Model(vgg16_model.input, vgg_x)
vgg16_final_model.compile(loss = 'categorical_crossentropy', optimizer= 'adam', metric
s=['acc'])
dplot.model(vgg16_final_model, to_file='model.png',show_shapes=True, show_layer_names=Fa
lse)
```

```
# FORTH
number_of_epochs = 25
vgg16_filepath = "vgg_16_" +saved-model-{epoch:02d}-acc-{val_acc:.2f}.hdf5"

vgg_checkpoint = tf.keras.callbacks.ModelCheckpoint(vgg16_filepath, monitor='val_acc',
verbose=1, save_best_only=True, mode='max')

vgg_early_stopping = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=5)

vgg16_history = vgg16_final_model.fit(train_generator, epochs = number_of_epochs ,vali
dation_data = validation_generator,callbacks=[vgg_checkpoint,vgg_early_stopping],verbo
se=1)

save_history(vgg16_history, 'vgg16_model')
plot_accuracy_from_history(vgg16_history)
plot_loss_from_history(vgg16_history)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/
vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58889256/58889256 [==============================] - 1s 0us/step
Epoch 1/25
455/455 [==============================] - ETA: 8s - loss: 0.6528 - acc: 0.7767
Epoch 00001: val_acc improved from inf to 0.85972, saving model to vgg_16_+saved-m
odel-01-acc-0.86.hdf5
455/455 [==============================] - 111s 250ms/step - loss: 0.6528 - acc: 0.
7767 - val_loss: 0.4423 - val_acc: 0.8597
Epoch 2/25
455/455 [==============================] - ETA: 8s - loss: 0.4325 - acc: 0.8411
Epoch 00002: val_acc improved from 0.85972 to 0.87673, saving model to vgg_16_+save
d-model-02-acc-0.88.hdf5
455/455 [==============================] - 108s 242ms/step - loss: 0.4325 - acc: 0.
8411 - val_loss: 0.3404 - val_acc: 0.8767
Epoch 3/25
455/455 [==============================] - ETA: 8s - loss: 0.3507 - acc: 0.8561
Epoch 00003: val_acc improved from 0.87673 to 0.88788, saving model to vgg_16_+save
d-model-03-acc-0.87.hdf5
455/455 [==============================] - 105s 230ms/step - loss: 0.3507 - acc: 0.
8561 - val_loss: 0.3015 - val_acc: 0.8879
Epoch 4/25
455/455 [==============================] - ETA: 8s - loss: 0.3791 - acc: 0.8613
Epoch 00004: val_acc did not improve from 0.88788
455/455 [==============================] - 101s 229ms/step - loss: 0.3791 - acc: 0.
8613 - val_loss: 0.3944 - val_acc: 0.8488
Epoch 5/25
455/455 [==============================] - ETA: 8s - loss: 0.3645 - acc: 0.8664
Epoch 00005: val_acc did not improve from 0.88788
455/455 [==============================] - 107s 240ms/step - loss: 0.3645 - acc: 0.
8664 - val_loss: 0.3009 - val_acc: 0.8839
Epoch 6/25
455/455 [==============================] - ETA: 8s - loss: 0.3484 - acc: 0.8748
Epoch 00006: val_acc improved from 0.88788 to 0.87673, saving model to vgg_16_+save
d-model-06-acc-0.87.hdf5
455/455 [==============================] - 105s 245ms/step - loss: 0.3484 - acc: 0.
8748 - val_loss: 0.3435 - val_acc: 0.8767
Epoch 7/25
455/455 [==============================] - ETA: 8s - loss: 0.3410 - acc: 0.8759
Epoch 00007: val_acc improved from 0.87673 to 0.87457, saving model to vgg_16_+save
d-model-07-acc-0.87.hdf5
455/455 [==============================] - 104s 227ms/step - loss: 0.3410 - acc: 0.
8759 - val_loss: 0.3375 - val_acc: 0.8747
Epoch 8/25
455/455 [==============================] - ETA: 8s - loss: 0.3306 - acc: 0.8827
Epoch 00008: val_acc did not improve from 0.87457
455/455 [==============================] - 105s 241ms/step - loss: 0.3306 - acc: 0.
8827 - val_loss: 0.3485 - val_acc: 0.8668
Epoch 9/25
455/455 [==============================] - ETA: 8s - loss: 0.3259 - acc: 0.8806
Epoch 00009: val_acc did not improve from 0.87457
455/455 [==============================] - 102s 230ms/step - loss: 0.3259 - acc: 0.
8806 - val_loss: 0.3304 - val_acc: 0.8655
Epoch 10/25
455/455 [==============================] - ETA: 8s - loss: 0.3157 - acc: 0.8814
Epoch 00010: val_acc did not improve from 0.87457
455/455 [==============================] - 108s 240ms/step - loss: 0.3157 - acc: 0.
8814 - val_loss: 0.3546 - val_acc: 0.8592
Epoch 11/25
455/455 [==============================] - ETA: 8s - loss: 0.3128 - acc: 0.8858
Epoch 00011: val_acc did not improve from 0.87457
455/455 [==============================] - 111s 252ms/step - loss: 0.3128 - acc: 0.
8858 - val_loss: 0.3585 - val_acc: 0.8548
Epoch 12/25
455/455 [==============================] - ETA: 8s - loss: 0.3060 - acc: 0.8868
Epoch 00012: val_acc did not improve from 0.87457
455/455 [==============================] - 109s 240ms/step - loss: 0.3060 - acc: 0.
8868 - val_loss: 0.3485 - val_acc: 0.8657
Epoch 13/25
455/455 [==============================] - ETA: 8s - loss: 0.2968 - acc: 0.8896
Epoch 00013: val_acc did not improve from 0.87457
455/455 [==============================] - 111s 252ms/step - loss: 0.2968 - acc: 0.
8896 - val_loss: 0.3311 - val_acc: 0.8742
Epoch 14/25
455/455 [==============================] - ETA: 8s - loss: 0.2977 - acc: 0.8879
Epoch 00014: val_acc did not improve from 0.87457
455/455 [==============================] - 109s 240ms/step - loss: 0.2977 - acc: 0.
8879 - val_loss: 0.3485 - val_acc: 0.8580
Epoch 15/25
455/455 [==============================] - ETA: 8s - loss: 0.2926 - acc: 0.8909
Epoch 00015: val_acc did not improve from 0.87457
455/455 [==============================] - 112s 251ms/step - loss: 0.2926 - acc: 0.
8909 - val_loss: 0.3386 - val_acc: 0.8655
Epoch 16/25
455/455 [==============================] - ETA: 8s - loss: 0.2851 - acc: 0.8914
Epoch 00016: val_acc improved from 0.87457 to 0.87853, saving model to vgg_16_+save
d-model-16-acc-0.88.hdf5
455/455 [==============================] - 112s 250ms/step - loss: 0.2851 - acc: 0.
8914 - val_loss: 0.3294 - val_acc: 0.8785
Epoch 17/25
455/455 [==============================] - ETA: 8s - loss: 0.2818 - acc: 0.8979
Epoch 00017: val_acc did not improve from 0.87853
455/455 [==============================] - 109s 240ms/step - loss: 0.2818 - acc: 0.
8979 - val_loss: 0.3389 - val_acc: 0.8747
Epoch 18/25
455/455 [==============================] - ETA: 8s - loss: 0.2769 - acc: 0.8986
Epoch 00018: val_acc did not improve from 0.87853
455/455 [==============================] - 110s 250ms/step - loss: 0.2769 - acc: 0.
8986 - val_loss: 0.3511 - val_acc: 0.8657
Epoch 19/25
455/455 [==============================] - ETA: 8s - loss: 0.2695 - acc: 0.9004
Epoch 00019: val_acc did not improve from 0.87853
455/455 [==============================] - 109s 240ms/step - loss: 0.2695 - acc: 0.
9004 - val_loss: 0.3373 - val_acc: 0.8747
Epoch 20/25
455/455 [==============================] - ETA: 8s - loss: 0.2711 - acc: 0.8999
Epoch 00020: val_acc did not improve from 0.87853
455/455 [==============================] - 109s 240ms/step - loss: 0.2711 - acc: 0.
8999 - val_loss: 0.3206 - val_acc: 0.8767
Epoch 21/25
455/455 [==============================] - ETA: 8s - loss: 0.2579 - acc: 0.9032
Epoch 00021: val_acc did not improve from 0.87853
455/455 [==============================] - 107s 240ms/step - loss: 0.2579 - acc: 0.
9032 - val_loss: 0.3001 - val_acc: 0.8868
Epoch 22/25
455/455 [==============================] - ETA: 8s - loss: 0.2540 - acc: 0.9069
Epoch 00022: val_acc did not improve from 0.87853
455/455 [==============================] - 110s 250ms/step - loss: 0.2540 - acc: 0.
9069 - val_loss: 0.3369 - val_acc: 0.8728
Epoch 23/25
455/455 [==============================] - ETA: 8s - loss: 0.2558 - acc: 0.9024
Epoch 00023: val_acc did not improve from 0.87853
455/455 [==============================] - 107s 240ms/step - loss: 0.2558 - acc: 0.
9024 - val_loss: 0.3439 - val_acc: 0.8603
Epoch 24/25
455/455 [==============================] - ETA: 8s - loss: 0.2531 - acc: 0.9079
Epoch 00024: val_acc did not improve from 0.87853
455/455 [==============================] - 111s 250ms/step - loss: 0.2532 - acc: 0.
9079 - val_loss: 0.3259 - val_acc: 0.8746
Epoch 25/25
455/455 [==============================] - ETA: 8s - loss: 0.2459 - acc: 0.9096
Epoch 00025: val_acc did not improve from 0.87853
455/455 [==============================] - 107s 240ms/step - loss: 0.2459 - acc: 0.
9096 - val_loss: 0.3458 - val_acc: 0.8653
```

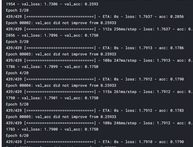
```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

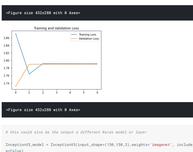
In [51]:

```
ResNet50_model = ResNet50(weights='imagenet', include_top=False, input_shape=(150,150,
3))

for layers in ResNet50_model.layers:
    layers.trainable=False

opt = SGD(lr=0.01,momentum=0.7)
resnet50_x = Conv2D(256, (3, 3), activation='relu')(ResNet50_model.output)
resnet50_x = MaxPooling2D(pool_size=(3, 3))(resnet50_x)
resnet50_x = Flatten()(resnet50_x)
resnet50_x = Dense(128,activation='relu')(resnet50_x)
resnet50_x = Dense(3,activation='softmax')(resnet50_x)

resnet50_x_final_model = Model(inputs=ResNet50_model.input, outputs=resnet50_x)
resnet50_x_final_model.compile(loss = 'categorical_crossentropy', optimizer= opt, metr
ics=['acc'])
dplot.model(resnet50_x_final_model, to_file='model.png',show_shapes=True, show_layer_name
s=True)
number_of_epochs = 20
resnet_filepath = "resnet50 +saved-model-{epoch:02d}-val_acc:{val_acc:.2f}.hdf5"

resnet_checkpoint = tf.keras.callbacks.ModelCheckpoint(resnet_filepath, monitor='val_a
cc', verbose=1, save_best_only=True, mode='max')

resnet_early_stopping = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=5)

resnet50_history = resnet50_x_final_model.fit(train_generator, epochs = number_of_epoc
hs ,validation_data = validation_generator,callbacks=[resnet_checkpoint,resnet_early_s
topping],verbose=1)

save_history(resnet50_history, 'resnet50_model')
plot_accuracy_from_history(resnet50_history)
plot_loss_from_history(resnet50_history)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/
resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94765736/94765736 [==============================] - 2s 0us/step
Epoch 1/20
455/455 [==============================] - ETA: 8s - loss: 1.8719 - acc: 0.7052
Epoch 00001: val_acc did not improve from inf
455/455 [==============================] - 110s 241ms/step - loss: 1.8719 - acc: 0.
7052 - val_loss: 1.7396 - val_acc: 0.2350
Epoch 2/20
455/455 [==============================] - ETA: 8s - loss: 1.7627 - acc: 0.2056
Epoch 00002: val_acc did not improve from 0.23503
455/455 [==============================] - 108s 237ms/step - loss: 1.7627 - acc: 0.
2056 - val_loss: 1.7396 - val_acc: 0.2350
Epoch 3/20
455/455 [==============================] - ETA: 8s - loss: 1.7753 - acc: 0.1730
Epoch 00003: val_acc did not improve from 0.23503
455/455 [==============================] - 108s 237ms/step - loss: 1.7753 - acc: 0.
1730 - val_loss: 1.7399 - val_acc: 0.2350
Epoch 4/20
455/455 [==============================] - ETA: 8s - loss: 1.7910 - acc: 0.1730
Epoch 00004: val_acc did not improve from 0.23503
455/455 [==============================] - 110s 241ms/step - loss: 1.7910 - acc: 0.
1730 - val_loss: 1.7401 - val_acc: 0.2350
Epoch 5/20
455/455 [==============================] - ETA: 8s - loss: 1.7971 - acc: 0.1777
Epoch 00005: val_acc did not improve from 0.23503
455/455 [==============================] - 107s 240ms/step - loss: 1.7971 - acc: 0.
1777 - val_loss: 1.7399 - val_acc: 0.2350
```




```
<Figure size 432x288 with 0 Axes>
```




```
<Figure size 432x288 with 0 Axes>
```

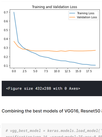
In [54]:

```
# this could also be the output a different Keras model or layer

Inception3_model = Inception3(input_shape=(150,150,3),weights='imagenet', include_to
p=False)
for layers in Inception3_model.layers[:299]:
    layers.trainable = False
for layer in Inception3_model.layers[299:]:
    layer.trainable = True
Inception3_last_output = Inception3_model.output
Inception3_mapped_output = Flatten()(Inception3_last_output)
Inception3_x = Dense(2056, activation='relu')(Inception3_mapped_output)
Inception3_x = Dropout(0.5)(Inception3_x)
Inception3_x = Dense(1024, activation='relu')(Inception3_mapped_output)
# and a logistic layer -- 3 classes
Inception3_x = Dense(3, activation='softmax')(Inception3_x)
Inception3_final_model = Model(inputs=Inception3_model.input,outputs=Inception3_
x)
Inception3_x_final_model.compile(optimizer=SGD(lr=0.0001, momentum=0.9), loss='catego
rical_crossentropy',metrics=['accuracy'])
dplot.model(Inception3_x_final_model, to_file='model.png',show_shapes=True, show_layer_
names=True)

number_of_epochs = 20
inception_filepath = "inception3_" +saved-model-{epoch:02d}-loss:{loss:.2f}.hdf5"

inception_checkpoint = tf.keras.callbacks.ModelCheckpoint(inception_filepath, monitor
='acc', verbose=1, save_best_only=True, mode='max')

inception_early_stopping = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=
5)

#inception_best_model = keras.models.load_model('./inception3+saved-models-on-intel-im
age-classification/inception.{epoch:02d}-{val_loss:2d}.h5')
inception3_history = Inception3_x_final_model.fit(train_generator, epochs = number_o
f_epochs, validation_data = validation_generator,callbacks=[inception_checkpoint,incep
tion_early_stopping],verbose=1)

save_history(inception3_history, 'inception3_model')
# we don't have accuracy information for inception3
plot_accuracy_from_history(inception3_history,True)
plot_loss_from_history(inception3_history)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/
inception_v3/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5
87910968/87910968 [==============================] - 1s 0us/step
Epoch 1/20
455/455 [==============================] - 105s 249ms/step - loss: 0.6999 - accuracy
: 0.7650 - val_loss: 0.3773 - val_accuracy: 0.8733
Epoch 2/20
455/455 [==============================] - 114s 250ms/step - loss: 0.3532 - accuracy
: 0.8747 - val_loss: 0.2180 - val_accuracy: 0.8917
Epoch 3/20
455/455 [==============================] - 105s 230ms/step - loss: 0.3088 - accuracy
: 0.8923 - val_loss: 0.2847 - val_accuracy: 0.8967
Epoch 4/20
455/455 [==============================] - 105s 247ms/step - loss: 0.2779 - accuracy
: 0.9005 - val_loss: 0.2010 - val_accuracy: 0.9043
Epoch 5/20
455/455 [==============================] - 105s 247ms/step - loss: 0.2577 - accuracy
: 0.9075 - val_loss: 0.2048 - val_accuracy: 0.9048
Epoch 6/20
455/455 [==============================] - 105s 248ms/step - loss: 0.2342 - accuracy
: 0.9156 - val_loss: 0.2625 - val_accuracy: 0.8927
Epoch 7/20
455/455 [==============================] - 105s 248ms/step - loss: 0.2078 - accuracy
: 0.9270 - val_loss: 0.2034 - val_accuracy: 0.9207
Epoch 8/20
455/455 [==============================] - 105s 248ms/step - loss: 0.1963 - accuracy
: 0.9302 - val_loss: 0.2346 - val_accuracy: 0.9053
Epoch 9/20
455/455 [==============================] - 105s 249ms/step - loss: 0.1898 - accuracy
: 0.9324 - val_loss: 0.2439 - val_accuracy: 0.9035
Epoch 10/20
455/455 [==============================] - 110s 251ms/step - loss: 0.1718 - accuracy
```

```
Epoch 12/20
435/435 [==============================] - 111s 254ms/step - loss: 0.7573 - accuracy:
0.9450 - val_loss: 0.2573 - val_accuracy: 0.9063
Epoch 13/20
435/435 [==============================] - 109s 249ms/step - loss: 0.7328 - accuracy:
0.9461 - val_loss: 0.2463 - val_accuracy: 0.9077
Epoch 14/20
435/435 [==============================] - 106s 237ms/step - loss: 0.7219 - accuracy:
0.9473 - val_loss: 0.2596 - val_accuracy: 0.9127
Epoch 15/20
435/435 [==============================] - 109s 247ms/step - loss: 0.7109 - accuracy:
0.9519 - val_loss: 0.2675 - val_accuracy: 0.9080
Epoch 16/20
435/435 [==============================] - 109s 249ms/step - loss: 0.7253 - accuracy:
0.9510 - val_loss: 0.2637 - val_accuracy: 0.9087
Epoch 17/20
435/435 [==============================] - 110s 251ms/step - loss: 0.7133 - accuracy:
0.9515 - val_loss: 0.2561 - val_accuracy: 0.9097
Epoch 18/20
435/435 [==============================] - 109s 248ms/step - loss: 0.7117 - accuracy:
0.9515 - val_loss: 0.2643 - val_accuracy: 0.9059
Epoch 19/20
435/435 [==============================] - 110s 250ms/step - loss: 0.7043 - accuracy:
0.9534 - val_loss: 0.2660 - val_accuracy: 0.9103
Epoch 20/20
435/435 [==============================] - 109s 249ms/step - loss: 0.6951 - accuracy:
0.9544 - val_loss: 0.2790 - val_accuracy: 0.9103
```

```
<Figure size 432x288 with 0 Axes>
```




```
<Figure size 432x288 with 0 Axes>
```

Combining the best models of VGG16, Resnet50 & Inception3

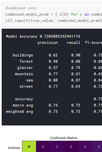
In [18]:
```
# vgg_best_model = keras.models.load_model('./input/pretrained-models-on-intel-image-cl
assification/vgg_16_cased_model-20-acc-0.84.h5')
# resnet_best_model = keras.models.load_model('./input/pretrained-models-on-intel-image
-classification/resnet50_cased_model-40-acc-0.82.h5')
# inception_best_model = keras.models.load_model('./input/pretrained-models-on-intel-im
age-classification/inception3_cased_model-30-loss-0.28.h5')

vgg_best_model = vgg16_final_model
resnet_best_model = resnet50_v_final_model
inception_best_model = Inception3_v_final_model
```

In [19]:
```
def models(y_list):
    ct = Counter(y_list)
    max_value = max(ct.values())
    return [key for key, value in ct.items() if value == max_value]

true_value = []
combined_model_pred = []
vgg_pred = []
resnet_pred = []
inception_pred = []
benchmark_model_pred = []
for folder in seg_test_folders:

    test_image_ids = os.listdir(os.path.join(root_path,'train_pred_test_folder',f'{i}'ras
n_pred_test_folders[i],folder))

    for image_id in test_image_ids[:int(len(test_image_ids)*0)]:

        path = os.path.join(root_path,'train_pred_test_folder',f'{i}',train_pred_test_folde
r[i],folder,image_id)

        true_value.append(validation_generator.class_indices[folder])
        img = cv2.resize(cv2.imread(path),(150,150))

        #vgg
        vgg16_image_prediction = np.argmax(vgg_best_model.predict(np.array([img])))
        vgg_pred.append(vgg16_image_prediction)

        #resnet50
        resnet_50_image_prediction = np.argmax(resnet_best_model.predict(np.array([im
g])))
        resnet_pred.append(resnet_50_image_prediction)

        #Inception
        inception_image_prediction = np.argmax(inception_best_model.predict(np.array
([img])))
        inception_pred.append(inception_image_prediction)

        #benchmark
        benchmark_model_prediction = np.argmax(benchmark_model.predict(np.array([im
g])))
        benchmark_model_pred.append(benchmark_model_prediction)

        #giving vgg16 high priority if they all predict something different
        image_prediction = models([vgg16_image_prediction, resnet_50_image_prediction, i
nception_image_prediction])
        combined_model_pred.append(image_prediction)
```

In [20]:
```
from sklearn.metrics import confusion_matrix
import textwrap
from matplotlib.ticker import plot_confusion_matrix
def clf_report(true_value, model_pred):
    classes = validation_generator.class_indices.keys()
    TP_count = [true_value[i] == model_pred[i] for i in range(len(true_value))]
    model_accuracy = np.sum(TP_count)/len(TP_count)
    print('Model Accuracy', model_accuracy)

    plt.figure(figsize=(7,7))
    cm = confusion_matrix(true_value,model_pred)
    plt.imshow(cm,interpolation='nearest',cmap=plt.cm.viridis)
    plt.title('Confusion Matrix')
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)
    thresh = cm.max()/10.0
    for i,j in itertools.product(range(cm.shape[0]),range(cm.shape[1])):
        plt.text(j,i,cm[i,j],
                 horizontalalignment='center',
                 color='black' if cm[i,j] > thresh else 'white')
        pass

    plt.ylabel('True Label')
    plt.xlabel('Predicted Label')
    pass

    print(classification_report(true_value, model_pred, target_names = list(classes)))
```

In [21]:
```
clf_report(true_value, benchmark_model_pred)
```

```
Model Accuracy 0.5895774255136
              precision    recall  f1-score   support

   buildings       0.50      0.73      0.60       189
      forest       0.96      0.65      0.81       118
    glacier       0.46      0.67      0.55       138
   mountain       0.66      0.31      0.42       131
        sea       0.61      0.94      0.77       127
     street       0.90      0.62      0.61       123

   accuracy                           0.59       746
  macro avg       0.62      0.65      0.59       746
weighted avg       0.62      0.59      0.59       746
```


Confusion Matrix
buildings
forest
glacier
mountain
sea
street
True Label
Predicted Label
138
2
51
1
0
0
0
116
0
0
0
0
0
0
93
0
0
0
0
0
0
123
0
0
0
0
0
0
80
0
0
0
0
0
0
76
0
0
0
0
0
0
77
0
0
0
0
0
0
76
0
0
0
0
0
0
76
0
0
0
0
0
0
76

In [27]:

```
show_few_images(f,combined_model,pred)
```


This Notebook has been released under the Apache 2.0 open source license.

Did you find this Notebook useful?
Show your appreciation with an upvote

Input

393.77 MB

Data Sources

- Intel Image Classification
  - seg_pred
    - seg_pred
  - seg_test
  - seg_train

Output

1.81 GB

- bench_mark_-model-01-0...
- bench_mark_-model-02-0...
- bench_mark_-model-03-0...
- bench_mark_-model-05-0...
- bench_mark_-model-06-0...
- bench_mark_-model-07-0...
- bench_mark_-model-08-0...
- bench_mark_-model-13-0...
- bench_mark_-model-14-0...
- bench_mark_-model-15-0...
- bench_mark_-model-18-0...
- bench_mark_-model_-model_lass...
- benchmark_model_history_...
- benchmark_model_history_...
- Xception3_model_history_...
- Xception3_model_history...
- resnet50-saved-model-01...
- resnet50_model_history.csv
- resnet50_model_history.json
- vgg16_model_history.csv
- vgg16_model_history.json

Download All

New Dataset

New Notebook

seg_pred (1 directories)

seg_pred
7301 files

bench_mark_-model-01-0.61.hdf5 (98.61 MB)

Unable to show preview

Previews for binary data are not supported

Execution Info

Succeeded	True	Run Time	6522.8 seconds
Exit Code	0	Timeout Exceeded	False
Used All Space	False	Output Size	1.81 GB
Environment	Container Image (Docker/Singularity)	Accelerator	GPU

Log

Download Log

```
Time  Line #  Log Message
2020-05-31    1    Model Accuracy 0.3092727102128
2020-05-31    2                precision    recall  f1-score   support
2020-05-31    3
2020-05-31    4    buildings     0.50      0.74      0.60       199
2020-05-31    5       forest     0.96      0.69      0.81       118
2020-05-31    6      glacier     0.36      0.07      0.12       139
2020-05-31    7     mountain     0.45      0.61      0.52       131
2020-05-31    8          sea     0.61      0.54      0.57       121
2020-05-31    9       street     0.68      0.82      0.61       125
2020-05-31   10
2020-05-31   11     accuracy                           0.59       748
2020-05-31   12    macro avg     0.63      0.59      0.59       748
2020-05-31   13 weighted avg     0.63      0.59      0.59       748
2020-05-31   14
2020-05-31   15 Model Accuracy 0.7585003329477196
2020-05-31   16                precision    recall  f1-score   support
2020-05-31   17
2020-05-31   18    buildings     0.63      0.56      0.73       199
2020-05-31   19       forest     0.90      0.86      0.88       118
2020-05-31   20      glacier     0.77      0.73      0.75       139
2020-05-31   21     mountain     0.77      0.77      0.80       131
2020-05-31   22          sea     0.80      0.87      0.84       121
2020-05-31   23       street     0.77      0.84      0.75       125
2020-05-31   24
2020-05-31   25     accuracy                           0.72       748
2020-05-31   26    macro avg     0.76      0.74      0.74       748
2020-05-31   27 weighted avg     0.75      0.72      0.71       748
2020-05-31   28
2020-05-31   29 Model Accuracy 0.7552631578947368
2020-05-31   30                precision    recall  f1-score   support
2020-05-31   31
2020-05-31   32    buildings     0.65      0.80      0.75       199
2020-05-31   33       forest     0.90      0.85      0.89       118
2020-05-31   34      glacier     0.70      0.80      0.67       139
2020-05-31   35     mountain     0.80      0.69      0.74       131
2020-05-31   36          sea     0.81      0.86      0.84       121
2020-05-31   37       street     0.77      0.73      0.75       125
2020-05-31   38
2020-05-31   39     accuracy                           0.74       748
2020-05-31   40    macro avg     0.76      0.74      0.73       748
2020-05-31   41 weighted avg     0.76      0.73      0.72       748
2020-05-31   42
2020-05-31   43 Model Accuracy 0.7470588235294117
2020-05-31   44                precision    recall  f1-score   support
2020-05-31   45
2020-05-31   46    buildings     0.74      0.80      0.76       199
2020-05-31   47       forest     1.00      0.81      0.89       118
2020-05-31   48      glacier     0.60      0.80      0.68       139
2020-05-31   49     mountain     0.80      0.80      0.80       131
2020-05-31   50          sea     0.88      0.80      0.84       121
2020-05-31   51       street     0.80      0.80      0.80       125
2020-05-31   52
2020-05-31   53     accuracy                           0.15       748
2020-05-31   54    macro avg     0.16      0.17      0.85       748
2020-05-31   55 weighted avg     0.16      0.15      0.84       748
2020-05-31   56
2020-05-31   57 /usr/local/lib/python3.6/dist-packages/sklearn/metrics/classification.py:1221:
               UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in
               labels with no predicted samples. Use `zero_division` parameter to control this
               behavior.
               _warn_prf(average, modifier, msg_start, len(result))
2020-05-31   58 Model Accuracy 0.8333333333333333
2020-05-31   59                precision    recall  f1-score   support
2020-05-31   60
2020-05-31   61
2020-05-31   62    buildings     0.17      0.80      0.83       199
2020-05-31   63       forest     0.17      0.83      0.15       118
2020-05-31   64      glacier     0.15      0.15      0.25       139
2020-05-31   65     mountain     0.14      0.11      0.12       131
2020-05-31   66          sea     0.25      0.57      0.43       121
2020-05-31   67       street     0.15      0.13      0.15       125
2020-05-31   68
2020-05-31   69     accuracy                           0.22       748
2020-05-31   70    macro avg     0.20      0.23      0.20       748
2020-05-31   71 weighted avg     0.20      0.22      0.20       748
2020-05-31   72
2020-05-31   73
2020-05-31   74 [NbConvertApp] Converting notebook __notebook__.ipynb to notebook
2020-05-31   75 [NbConvertApp] Writing 266070 bytes to __notebook__.ipynb
2020-05-31   76 [NbConvertApp] Converting notebook __notebook__.ipynb to html
2020-05-31   77 [NbConvertApp] Support files will be in __results___files/
2020-05-31   78 [NbConvertApp] Making directory __results___files
2020-05-31   79 [NbConvertApp] Making directory __results___files
2020-05-31   80 [NbConvertApp] Making directory __results___files
2020-05-31   81 [NbConvertApp] Making directory __results___files
2020-05-31   82 [NbConvertApp] Making directory __results___files
2020-05-31   83 [NbConvertApp] Making directory __results___files
2020-05-31   84 [NbConvertApp] Making directory __results___files
2020-05-31   85 [NbConvertApp] Making directory __results___files
2020-05-31   86 [NbConvertApp] Making directory __results___files
2020-05-31   87 [NbConvertApp] Making directory __results___files
2020-05-31   88 [NbConvertApp] Making directory __results___files
2020-05-31   89 [NbConvertApp] Making directory __results___files
2020-05-31   90 [NbConvertApp] Making directory __results___files
2020-05-31   91 [NbConvertApp] Making directory __results___files
2020-05-31   92 [NbConvertApp] Making directory __results___files
2020-05-31   93 [NbConvertApp] Making directory __results___files
2020-05-31   94 [NbConvertApp] Making directory __results___files
2020-05-31   95 [NbConvertApp] Making directory __results___files
2020-05-31   96 [NbConvertApp] Writing 268004 bytes to __results___.html
2020-05-31   97
2020-05-31   98 Complete. Exited with code 0
```

Comments (0)

Sort by

Comment here. Be patient, be friendly, and focus on ideas. We're all here to learn and improve!

Post Comment