Team Id:PNT2022TMID46648

Model Building

Compile To The Model

In [ ]:

```
from tensorflow.keras.preprocessing.image  import
ImageDataGenerator
```

In [ ]:

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

In [ ]:

```
# Creating sample sourcecode to multiply two variables
# x and y.
srcCode = 'x = 10\ny = 20\nmul = x * y\nprint("mul =", mul)'

# Converting above source code to an executable execCode
= compile(srcCode, 'mulstring', 'exec')

# Running the executable code. exec(execCode)
```

In [ ]:

```
# Training Datagen train_datagen
=
ImageDataGenerator(rescale=1/255,zoom_range=0.2,horizontal_flip=True,vertica
l_flip=False) # Testing Datagen
test_datagen = ImageDataGenerator(rescale=1/255)
```

In [ ]:

```
# Training Dataset
x_train=train_datagen.flow_from_directory(r'/content/drive/MyDrive/Dataset/t
raining_set',target_size=(64,64), class_mode='categorical',batch_size=900)
# Testing Dataset
x_test=test_datagen.flow_from_directory(r'/content/drive/MyDrive/Dataset/tes
t_set',target_size=(64,64), class_mode='categorical',batch_size=900)
```

```
Found 15760 images belonging to 9 classes.
Found 2250 images belonging to 9 classes.
```

In [ ]:

```
def compile_model_results(model, root="./"):
     listing = glob.glob(root + '/models/' + model +
'/*/best_pars.pkl')
     dic_list = []
for file in listing:
        tmp = hyper_parameters_load(file)
dic_list.append(tmp.to_dictionary())      df = pd.DataFrame(dic_list)
df['diff'] = df.test_F1 - df.forecast_F1     df['pci'] = abs(df.test_F1 -
df.forecast_F1)
```

```
    if not os.path.exists(root + '/figures/' +  model
):
        os.makedirs(root + '/figures/' +  model )
    df.to_csv(root + '/figures/' +  model + '/results.csv',
index=False)
    return
df
```

In [ ]:

```
# Set optimizer loss and metrics     opt = Adam(lr=args.initial_lr,
beta_1=0.99, beta_2=0.999, decay=1e-6)     if args.net.find('caps') != -
1:
        metrics = {'out_seg': dice_hard}
else:
        metrics = [dice_hard]
    loss, loss_weighting = get_loss(root=args.data_root_dir,
split=args.split_num,                              net=args.net,
recon_wei=args.recon_wei, choice=args.loss)

    # If using CPU or single GPU
if args.gpus <= 1:
        uncomp_model.compile(optimizer=opt, loss=loss,
loss_weights=loss_weighting, metrics=metrics)
return uncomp_model     # If using multiple GPUs
else:         with tf.device("/cpu:0"):
            uncomp_model.compile(optimizer=opt, loss=loss,
loss_weights=loss_weighting, metrics=metrics)             model =
multi_gpu_model(uncomp_model, gpus=args.gpus)
model.__setattr__('callback_model', uncomp_model)
model.compile(optimizer=opt, loss=loss, loss_weights=loss_weighting,
metrics=metrics)

X = array[:,0:8] Y
= array[:,8]
test_size = 0.33
seed = 7
X_train, X_test, Y_train, Y_test = model_selection.train_test_split(X, Y,
test_size=test_size,  random_state=seed)
```

In [ ]:

```
print("Len x-train : ", len(x_train)) print("Len
x-test : ", len(x_test))
```

```
Len x-train :  18
Len x-test :  3
```

In [ ]:

```
# The Class Indices in Training Dataset
x_train.class_indices
```

Out[ ]:

```
{'A': 0, 'B': 1, 'C': 2, 'D': 3, 'E': 4, 'F': 5, 'G': 6, 'H': 7, 'I': 8}
```

Model Compilation

In [ ]:

```
# Importing Libraries from
tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Convolution2D,MaxPooling2D,Flatten,Dense
```

In [ ]:

```
# Creating Model model=Sequential()
```

In [ ]:

```
# Adding Layers
model.add(Convolution2D(32,(3,3),activation='relu',input_shape=(64,64,3)))
```

In [ ]:

```
model.add(MaxPooling2D(pool_size=(2,2))) model.add(Flatten())
```

In [ ]:

```
# Adding Dense Layers model.add(Dense(300,activation='relu'))
model.add(Dense(150,activation='relu'))
model.add(Dense(9,activation='softmax'))
```

In [ ]:

```
# Compiling the Model
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['acc
uracy'])
```

In [ ]:

```
# reading code from a file f
= open('main.py', 'r') temp
= f.read()
f.close()      code   =   compile(temp,
'main.py', 'exec') exec(code)
```

Saving the Model

In [ ]:

```
model.save('asl_model_84_54.h5
```