

VGG-16 Model

Action Classes - 10

```
In [1]: from keras import models
from keras.layers import Dense, Flatten
from keras import backend as K
import numpy as np
import matplotlib.pyplot as plt

from keras.applications import vgg16
```

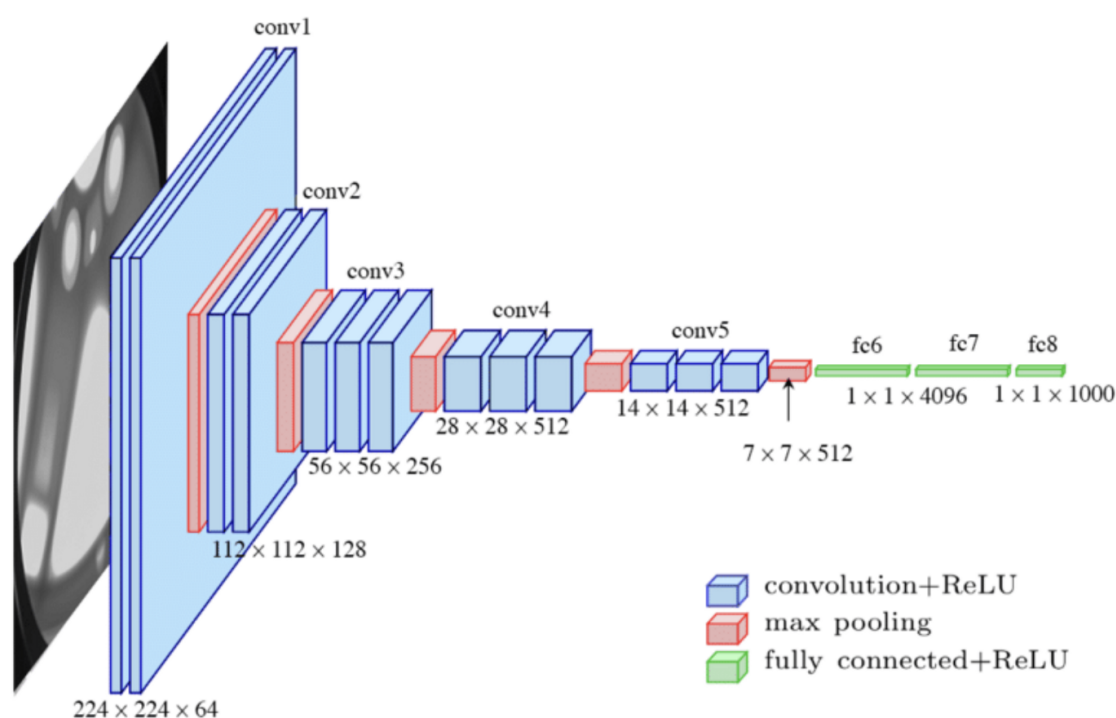
```
In [2]: import tensorflow as tf
print("Num GPUs Available: ", len(tf.config.list_physical_devices('GPU')))
```

Num GPUs Available: 1

2022-08-25 15:22:33.106107: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:975] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero

2022-08-25 15:22:33.222823: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:975] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero

2022-08-25 15:22:33.223112: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:975] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero



Dataset

```
In [3]: from keras.preprocessing.image import ImageDataGenerator

dataset_path = "./frames/"
# will contain the categories in respective folders

# Data generators
train_datagen = ImageDataGenerator(rescale=1/255, validation_split=0.
```

```
In [4]: image_size = (224,224)
batch_size = 10

train_batches = train_datagen.flow_from_directory(
    dataset_path,
    target_size = image_size,
    batch_size = batch_size,
    class_mode = "categorical",
    subset = "training"
)

validation_batches = train_datagen.flow_from_directory(
    dataset_path,
    target_size = image_size,
    batch_size = batch_size,
    class_mode = "categorical",
    subset = "validation"
)

test_batches = train_datagen.flow_from_directory(
    dataset_path,
    target_size = image_size,
    batch_size = batch_size,
    class_mode = "categorical",
    subset = "validation"
)
```

Found 2734 images belonging to 10 classes.
Found 679 images belonging to 10 classes.
Found 679 images belonging to 10 classes.

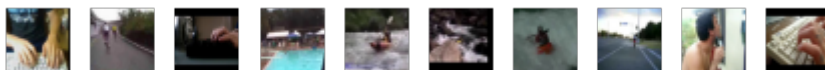
```
In [5]: train_batches.class_indices
```

```
Out[5]: {'ApplyLipstick': 0,
        'Archery': 1,
        'Biking': 2,
        'Diving': 3,
        'Kayaking': 4,
        'MilitaryParade': 5,
        'ShavingBeard': 6,
        'SkateBoarding': 7,
        'TennisSwing': 8,
        'Typing': 9}
```

```
In [6]: from matplotlib import pyplot as plt

def plot_images(images_arr):
    fig, axes = plt.subplots(1,10)
    axes = axes.flatten()
    for img, ax in zip(images_arr, axes):
        ax.imshow(img)
        ax.axis('off')
    plt.tight_layout()
    plt.show()
```

```
In [7]: imgs, labels = train_batches[0]
plot_images(imgs)
print(labels[:10])
```



```
[[0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]]
```

Initialize Model

```
In [8]: vggmodeltop = vgg16.VGG16(include_top=True,
                                input_shape=(224,224,3),
                                pooling='avg',
                                weights='imagenet')

for (i,layer) in enumerate(vggmodeltop.layers):
    layer.trainable = False
    print((i, layer.name, layer.output_shape))
```

2022-08-25 15:22:34.228222: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations: AVX2 FMA

To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.

2022-08-25 15:22:34.229689: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:975] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero

2022-08-25 15:22:34.229951: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:975] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero

2022-08-25 15:22:34.230067: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:975] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero

2022-08-25 15:22:35.398069: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:975] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero

2022-08-25 15:22:35.398228: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:975] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero

2022-08-25 15:22:35.398352: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:975] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero

2022-08-25 15:22:35.398471: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1532] Created device /job:localhost/replica:0/task:0/device:GPU:0 with 3368 MB memory: -> device: 0, name: NVIDIA GeForce GTX 1050 Ti, pci bus id: 0000:01:00.0, compute capability: 6.1

```
(0, 'input_1', [(None, 224, 224, 3)])
(1, 'block1_conv1', (None, 224, 224, 64))
(2, 'block1_conv2', (None, 224, 224, 64))
(3, 'block1_pool', (None, 112, 112, 64))
(4, 'block2_conv1', (None, 112, 112, 128))
(5, 'block2_conv2', (None, 112, 112, 128))
(6, 'block2_pool', (None, 56, 56, 128))
(7, 'block3_conv1', (None, 56, 56, 256))
(8, 'block3_conv2', (None, 56, 56, 256))
(9, 'block3_conv3', (None, 56, 56, 256))
(10, 'block3_pool', (None, 28, 28, 256))
(11, 'block4_conv1', (None, 28, 28, 512))
(12, 'block4_conv2', (None, 28, 28, 512))
(13, 'block4_conv3', (None, 28, 28, 512))
(14, 'block4_pool', (None, 14, 14, 512))
(15, 'block5_conv1', (None, 14, 14, 512))
```

```
(16, 'block5_conv2', (None, 14, 14, 512))
(17, 'block5_conv3', (None, 14, 14, 512))
(18, 'block5_pool', (None, 7, 7, 512))
(19, 'flatten', (None, 25088))
(20, 'fc1', (None, 4096))
(21, 'fc2', (None, 4096))
(22, 'predictions', (None, 1000))
```

```
In [9]: vggmodel = vgg16.VGG16(include_top=False,
                                input_shape=(224,224,3),
                                pooling='avg',classes=10,
                                weights='imagenet')

for (i,layer) in enumerate(vggmodel.layers):
    layer.trainable = False
    print((i, layer.name, layer.output_shape))
```

```
(0, 'input_2', [(None, 224, 224, 3)])
(1, 'block1_conv1', (None, 224, 224, 64))
(2, 'block1_conv2', (None, 224, 224, 64))
(3, 'block1_pool', (None, 112, 112, 64))
(4, 'block2_conv1', (None, 112, 112, 128))
(5, 'block2_conv2', (None, 112, 112, 128))
(6, 'block2_pool', (None, 56, 56, 128))
(7, 'block3_conv1', (None, 56, 56, 256))
(8, 'block3_conv2', (None, 56, 56, 256))
(9, 'block3_conv3', (None, 56, 56, 256))
(10, 'block3_pool', (None, 28, 28, 256))
(11, 'block4_conv1', (None, 28, 28, 512))
(12, 'block4_conv2', (None, 28, 28, 512))
(13, 'block4_conv3', (None, 28, 28, 512))
(14, 'block4_pool', (None, 14, 14, 512))
(15, 'block5_conv1', (None, 14, 14, 512))
(16, 'block5_conv2', (None, 14, 14, 512))
(17, 'block5_conv3', (None, 14, 14, 512))
(18, 'block5_pool', (None, 7, 7, 512))
(19, 'global_average_pooling2d', (None, 512))
```

```
In [10]: model = models.Sequential()

dense_layer_1 = Dense(32, activation='relu')
dense_layer_2 = Dense(32, activation='relu')
prediction_layer = Dense(10, activation='softmax')

model.add(vggmodel)
model.add(dense_layer_1)
model.add(dense_layer_2)
model.add(prediction_layer)

model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
vgg16 (Functional)	(None, 512)	14714688
dense (Dense)	(None, 32)	16416
dense_1 (Dense)	(None, 32)	1056
dense_2 (Dense)	(None, 10)	330

```
=====
Total params: 14,732,490
Trainable params: 17,802
Non-trainable params: 14,714,688
```

```
In [11]: model.compile(
    optimizer='adam',
    loss='categorical_crossentropy',
    metrics=['accuracy'],
)
```

```
In [12]: model.save("./models/action-class-10-model-vgg16.h5")
```

```
In [13]: fit = model.fit(train_batches, epochs=20, validation_data=validation
```

Epoch 1/20

2022-08-25 15:22:40.516591: I tensorflow/stream_executor/cuda/cuda_dnn.cc:384] Loaded cuDNN version 8401

2022-08-25 15:22:42.065437: I tensorflow/core/platform/default/subprocess.cc:304] Start cannot spawn child process: No such file or directory

2022-08-25 15:22:42.377251: W tensorflow/core/common_runtime/bfc_allocator.cc:290] Allocator (GPU_0_bfc) ran out of memory trying to allocate 2.35GiB with freed_by_count=0. The caller indicates that this is not a failure, but this may mean that there could be performance gains if more memory were available.

274/274 [=====] - ETA: 0s - loss: 1.5440 - accuracy: 0.5117

2022-08-25 15:23:34.447861: W tensorflow/core/common_runtime/bfc_allocator.cc:290] Allocator (GPU_0_bfc) ran out of memory trying to allocate 2.31GiB with freed_by_count=0. The caller indicates that this is not a failure, but this may mean that there could be performance gains if more memory were available.

274/274 [=====] - 57s 190ms/step - loss: 1.5440 - accuracy: 0.5117 - val_loss: 1.0611 - val_accuracy: 0.7113
Epoch 2/20

274/274 [=====] - 48s 177ms/step - loss: 0.6726 - accuracy: 0.8175 - val_loss: 0.6954 - val_accuracy: 0.7599

Epoch 3/20

274/274 [=====] - 50s 184ms/step - loss: 0.4233 - accuracy: 0.8841 - val_loss: 0.5965 - val_accuracy: 0.7806

Epoch 4/20

274/274 [=====] - 66s 242ms/step - loss: 0.3103 - accuracy: 0.9115 - val_loss: 0.6744 - val_accuracy: 0.7688

Epoch 5/20

274/274 [=====] - 79s 287ms/step - loss: 0.2427 - accuracy: 0.9305 - val_loss: 0.5453 - val_accuracy: 0.8100

Epoch 6/20

274/274 [=====] - 79s 288ms/step - loss: 0.2033 - accuracy: 0.9396 - val_loss: 0.5187 - val_accuracy: 0.8056

Epoch 7/20

274/274 [=====] - 81s 293ms/step - loss: 0.1638 - accuracy: 0.9550 - val_loss: 0.4840 - val_accuracy: 0.8144

Epoch 8/20

274/274 [=====] - 81s 294ms/step - loss: 0.1318 - accuracy: 0.9682 - val_loss: 0.4937 - val_accuracy: 0.8130

Epoch 9/20

274/274 [=====] - 81s 294ms/step - loss: 0.1251 - accuracy: 0.9623 - val_loss: 0.4243 - val_accuracy: 0.8424

Epoch 10/20

274/274 [=====] - 81s 295ms/step - loss: 0.1044 - accuracy: 0.9737 - val_loss: 0.4737 - val_accuracy: 0.8247

Epoch 11/20

274/274 [=====] - 82s 297ms/step - loss: 0.0907 - accuracy: 0.9755 - val_loss: 0.4386 - val_accuracy: 0.8292

Epoch 12/20

274/274 [=====] - 81s 296ms/step - loss: 0.0797 - accuracy: 0.9795 - val_loss: 0.6962 - val_accuracy: 0.8159

Epoch 13/20

```

274/274 [=====] - 81s 295ms/step - loss:
0.0699 - accuracy: 0.9817 - val_loss: 0.5364 - val_accuracy: 0.8218
Epoch 14/20
274/274 [=====] - 81s 295ms/step - loss:
0.0608 - accuracy: 0.9857 - val_loss: 0.4659 - val_accuracy: 0.8336
Epoch 15/20
274/274 [=====] - 81s 296ms/step - loss:
0.0543 - accuracy: 0.9872 - val_loss: 0.4637 - val_accuracy: 0.8351
Epoch 16/20
274/274 [=====] - 81s 295ms/step - loss:
0.0483 - accuracy: 0.9883 - val_loss: 0.4740 - val_accuracy: 0.8306
Epoch 17/20
274/274 [=====] - 81s 294ms/step - loss:
0.0418 - accuracy: 0.9898 - val_loss: 0.4922 - val_accuracy: 0.8321
Epoch 18/20
274/274 [=====] - 81s 295ms/step - loss:
0.0368 - accuracy: 0.9938 - val_loss: 0.5208 - val_accuracy: 0.8247
Epoch 19/20
274/274 [=====] - 82s 297ms/step - loss:
0.0358 - accuracy: 0.9901 - val_loss: 0.5735 - val_accuracy: 0.8292
Epoch 20/20
274/274 [=====] - 82s 297ms/step - loss:
0.0306 - accuracy: 0.9945 - val_loss: 0.5131 - val_accuracy: 0.8306

```

```
In [14]: model.save("./models/action-class-10-trained-vgg16.h5")
```

Evaluate and Predict

```
In [15]: model = models.load_model("./models/action-class-10-trained-vgg16.h5")
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
vgg16 (Functional)	(None, 512)	14714688
dense (Dense)	(None, 32)	16416
dense_1 (Dense)	(None, 32)	1056
dense_2 (Dense)	(None, 10)	330

```

=====
Total params: 14,732,490
Trainable params: 17,802
Non-trainable params: 14,714,688

```

```
In [16]: model.evaluate(test_batches)
```

```

68/68 [=====] - 20s 224ms/step - loss: 0.5131 - accuracy: 0.8306

```

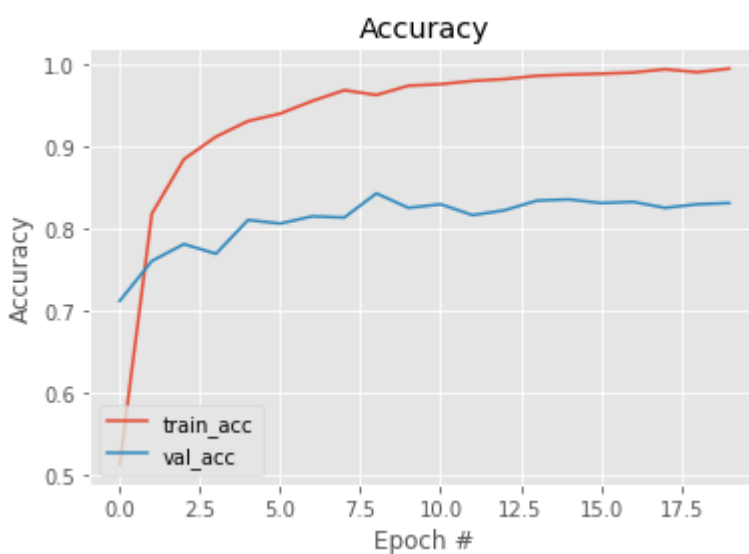
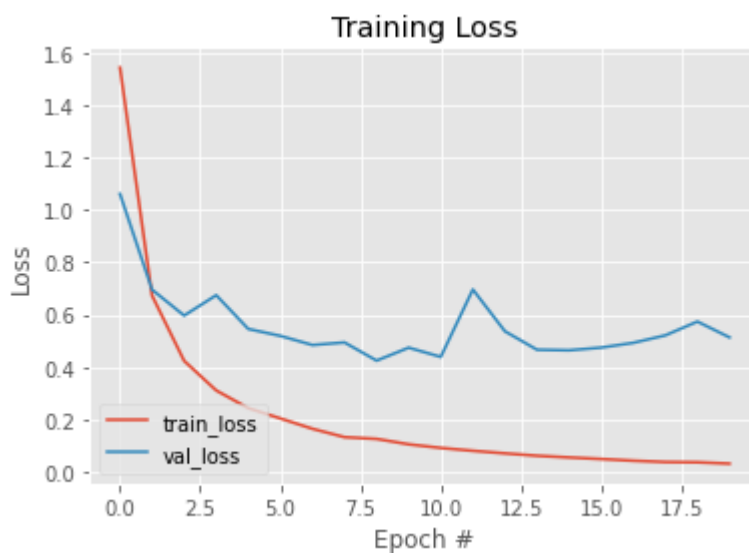
```
Out[16]: [0.5130749940872192, 0.830633282661438]
```



```
In [17]: plt.style.use("ggplot")
plt.figure()

plt.plot(np.arange(0, 20), fit.history["loss"], label="train_loss")
plt.plot(np.arange(0, 20), fit.history["val_loss"], label="val_loss")
plt.title("Training Loss")
plt.xlabel("Epoch #")
plt.ylabel("Loss")
plt.legend(loc="lower left")
plt.show()

plt.plot(np.arange(0, 20), fit.history["accuracy"], label="train_acc")
plt.plot(np.arange(0, 20), fit.history["val_accuracy"], label="val_acc")
plt.title("Accuracy")
plt.xlabel("Epoch #")
plt.ylabel("Accuracy")
plt.legend(loc="lower left")
plt.show()
```



```
In [18]: print("Avg Val Acc: " + str(sum(fit.history["val_accuracy"])/20))
print("Avg Val Loss: " + str(sum(fit.history["val_loss"])/20))
```

Avg Val Acc: 0.8106774628162384
Avg Val Loss: 0.5570720851421356

