VGG-16 Model

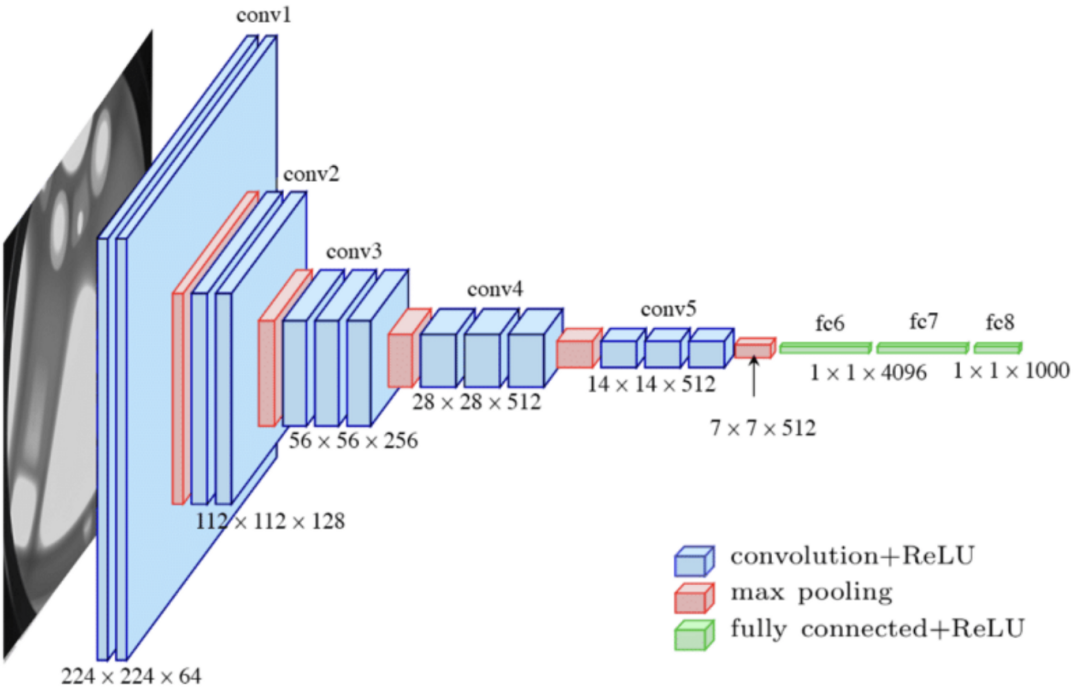
Action Classes - 5

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 22:55:33.765584: I tensorflow/stream_executor/cuda/cuda_gpu_ex
ecutor.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 22:55:33.881975: I tensorflow/stream_executor/cuda/cuda_gpu_ex
ecutor.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 22:55:33.882232: I tensorflow/stream_executor/cuda/cuda_gpu_ex
ecutor.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=0.2)
```

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 1546 images belonging to 5 classes.
Found 384 images belonging to 5 classes.
Found 384 images belonging to 5 classes.
```

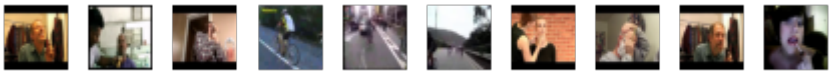
In [5]:
```
train_batches.class_indices
```

Out[5]:
```
{'ApplyLipstick': 0,
 'Biking': 1,
 'Kayaking': 2,
 'ShavingBeard': 3,
 'TennisSwing': 4}
```

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

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 22:55:35.024697: I tensorflow/core/platform/cpu_feature_guard.
cc:193] This TensorFlow binary is optimized with oneAPI Deep Neural Netwo
rk Library (oneDNN) to use the following CPU instructions in performance-
critical operations:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropria
te compiler flags.
2022-08-25 22:55:35.025857: I tensorflow/stream_executor/cuda/cuda_gpu_ex
ecutor.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 22:55:35.026183: I tensorflow/stream_executor/cuda/cuda_gpu_ex
ecutor.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 22:55:35.026463: I tensorflow/stream_executor/cuda/cuda_gpu_ex
ecutor.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 22:55:36.156319: I tensorflow/stream_executor/cuda/cuda_gpu_ex
ecutor.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 22:55:36.156469: I tensorflow/stream_executor/cuda/cuda_gpu_ex
ecutor.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 22:55:36.156582: I tensorflow/stream_executor/cuda/cuda_gpu_ex
ecutor.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 22:55:36.156683: I tensorflow/core/common_runtime/gpu/gpu_devi
ce.cc:1532] Created device /job:localhost/replica:0/task:0/device:GPU:0 w
ith 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=5,
                       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(5, 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, 5)                 165

=================================================================
Total params: 14,732,325
Trainable params: 17,637
Non-trainable params: 14,714,688
_________________________________________________________________
```

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

In [12]:
```
model.save("./models/action-class-05-vgg16.h5")
```

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

```
Epoch 1/20
2022-08-25 22:55:41.586350: I tensorflow/stream_executor/cuda/cuda_dnn.c
c:384] Loaded cuDNN version 8401
2022-08-25 22:55:43.017749: I tensorflow/core/platform/default/subproces
s.cc:304] Start cannot spawn child process: No such file or directory
2022-08-25 22:55:43.300315: W tensorflow/core/common_runtime/bfc_allocato
r.cc:290] Allocator (GPU_0_bfc) ran out of memory trying to allocate 2.35
GiB with freed_by_count=0. The caller indicates that this is not a failur
e, but this may mean that there could be performance gains if more memory
were available.
```

```
155/155 [==============================] - 34s 188ms/step - loss: 1.0169
- accuracy: 0.6397 - val_loss: 0.6873 - val_accuracy: 0.7734
Epoch 2/20
155/155 [==============================] - 27s 176ms/step - loss: 0.3839
- accuracy: 0.8829 - val_loss: 0.5256 - val_accuracy: 0.8047
Epoch 3/20
155/155 [==============================] - 28s 178ms/step - loss: 0.2397
- accuracy: 0.9288 - val_loss: 0.4590 - val_accuracy: 0.8333
Epoch 4/20
155/155 [==============================] - 28s 183ms/step - loss: 0.1712
- accuracy: 0.9483 - val_loss: 0.4079 - val_accuracy: 0.8359
Epoch 5/20
155/155 [==============================] - 29s 188ms/step - loss: 0.1240
- accuracy: 0.9664 - val_loss: 0.4187 - val_accuracy: 0.8516
Epoch 6/20
155/155 [==============================] - 30s 196ms/step - loss: 0.0893
- accuracy: 0.9754 - val_loss: 0.4471 - val_accuracy: 0.8516
Epoch 7/20
155/155 [==============================] - 40s 258ms/step - loss: 0.0772
- accuracy: 0.9774 - val_loss: 0.3868 - val_accuracy: 0.8698
Epoch 8/20
155/155 [==============================] - 45s 291ms/step - loss: 0.0667
- accuracy: 0.9858 - val_loss: 0.3802 - val_accuracy: 0.8724
Epoch 9/20
155/155 [==============================] - 45s 290ms/step - loss: 0.0501
- accuracy: 0.9877 - val_loss: 0.3738 - val_accuracy: 0.8776
Epoch 10/20
155/155 [==============================] - 46s 293ms/step - loss: 0.0443
- accuracy: 0.9890 - val_loss: 0.3991 - val_accuracy: 0.8594
Epoch 11/20
155/155 [==============================] - 46s 295ms/step - loss: 0.0348
- accuracy: 0.9916 - val_loss: 0.4635 - val_accuracy: 0.8568
Epoch 12/20
155/155 [==============================] - 46s 293ms/step - loss: 0.0340
- accuracy: 0.9929 - val_loss: 0.4530 - val_accuracy: 0.8568
Epoch 13/20
155/155 [==============================] - 46s 297ms/step - loss: 0.0274
- accuracy: 0.9955 - val_loss: 0.4592 - val_accuracy: 0.8281
Epoch 14/20
155/155 [==============================] - 45s 292ms/step - loss: 0.0230
- accuracy: 0.9961 - val_loss: 0.4194 - val_accuracy: 0.8568
Epoch 15/20
155/155 [==============================] - 47s 299ms/step - loss: 0.0233
- accuracy: 0.9942 - val_loss: 0.4471 - val_accuracy: 0.8646
Epoch 16/20
155/155 [==============================] - 46s 297ms/step - loss: 0.0170
- accuracy: 0.9981 - val_loss: 0.4513 - val_accuracy: 0.8672
Epoch 17/20
155/155 [==============================] - 46s 296ms/step - loss: 0.0162
- accuracy: 0.9974 - val_loss: 0.3773 - val_accuracy: 0.8828
Epoch 18/20
155/155 [==============================] - 46s 294ms/step - loss: 0.0127
- accuracy: 0.9981 - val_loss: 0.4037 - val_accuracy: 0.8776
Epoch 19/20
155/155 [==============================] - 47s 303ms/step - loss: 0.0124
- accuracy: 0.9974 - val_loss: 0.4426 - val_accuracy: 0.8620
Epoch 20/20
155/155 [==============================] - 46s 294ms/step - loss: 0.0117
- accuracy: 0.9974 - val_loss: 0.4558 - val_accuracy: 0.8646
```

In [14]:
```
model.save("./models/action-class-05-trained-vgg16.h5")
```

Evaluate and Predict

In [15]:
```
model = models.load_model("./models/action-class-05-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, 5)                 165

=================================================================
Total params: 14,732,325
Trainable params: 17,637
Non-trainable params: 14,714,688
_________________________________________________________________
```

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

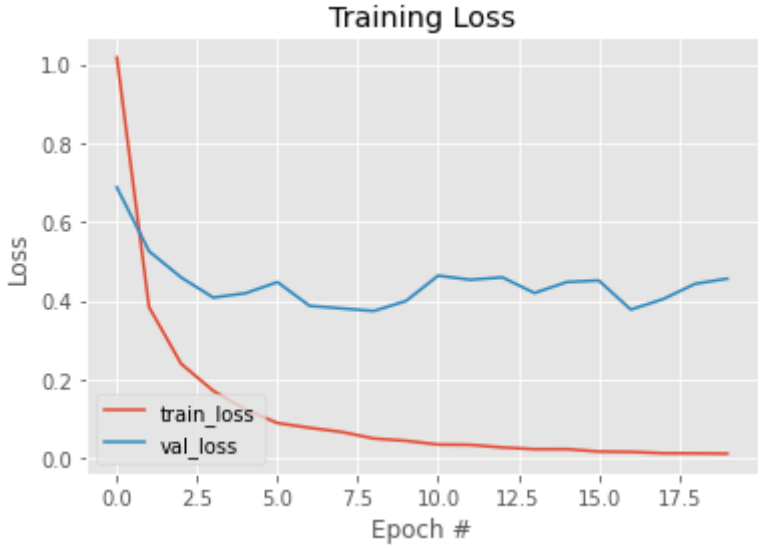
```
39/39 [==============================] - 13s 229ms/step - loss: 0.4558 -
accuracy: 0.8646
```

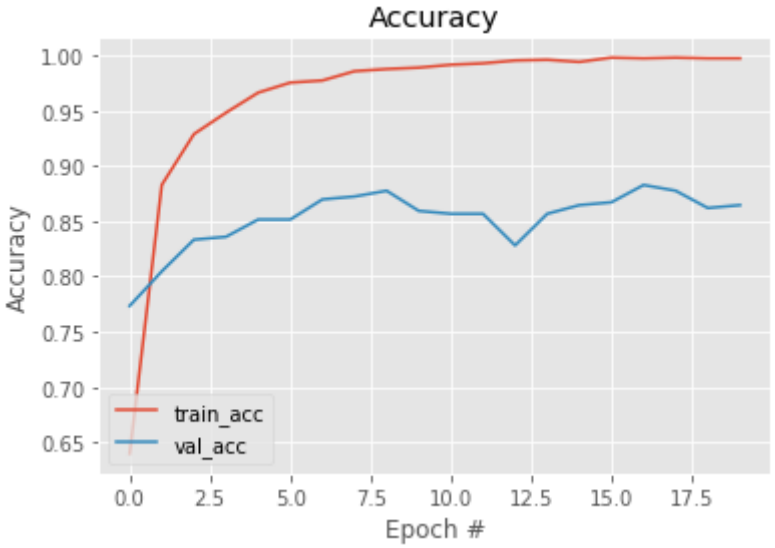
Out[16]:
```
[0.4558173418045044, 0.8645833134651184]
```

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*100))
print("Avg Val Loss: " + str(sum(fit.history["val_loss"])/20*100))
```

```
Avg Val Acc: 85.23437470197678
Avg Val Loss: 44.29270029067993
```