VGG-16 Model

Action Classes - 20

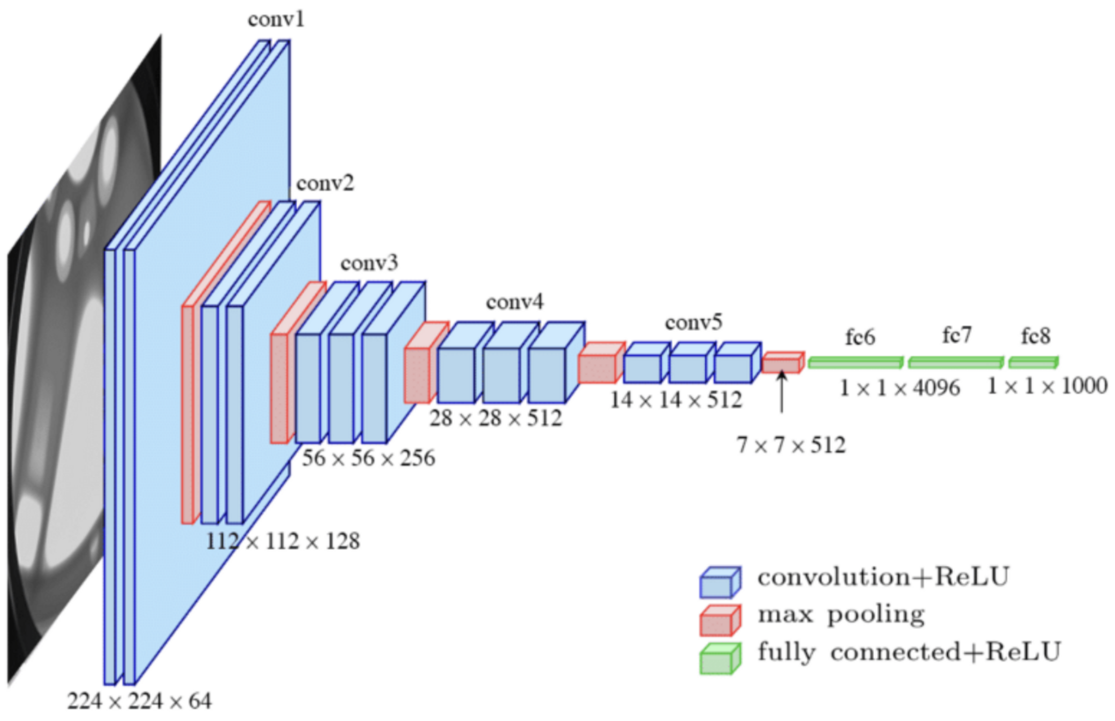
In [1]:
```
from keras import models
from keras.layers import Dense,Flatten
from keras import backend as K
import numpy as np
import matplotlib.pyplot as plt

from keras.applications import vgg16
```

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

```
Num GPUs Available:  1

2022-08-25 20:01:09.362085: I tensorflow/stream_executor/cuda/cuda_
gpu_executor.cc:975] successful NUMA node read from SysFS had negat
ive value (-1), but there must be at least one NUMA node, so return
ing NUMA node zero
2022-08-25 20:01:09.499601: I tensorflow/stream_executor/cuda/cuda_
gpu_executor.cc:975] successful NUMA node read from SysFS had negat
ive value (-1), but there must be at least one NUMA node, so return
ing NUMA node zero
2022-08-25 20:01:09.499857: I tensorflow/stream_executor/cuda/cuda_
gpu_executor.cc:975] successful NUMA node read from SysFS had negat
ive value (-1), but there must be at least one NUMA node, so return
ing 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 5118 images belonging to 20 classes.
Found 1270 images belonging to 20 classes.
Found 1270 images belonging to 20 classes.
```

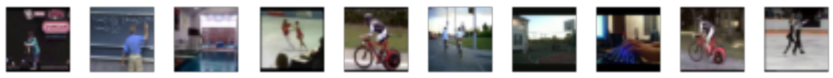
In [5]:
```
train_batches.class_indices
```

Out[5]:
```
{'ApplyLipstick': 0,
 'Archery': 1,
 'BabyCrawling': 2,
 'Basketball': 3,
 'Biking': 4,
 'Diving': 5,
 'Fencing': 6,
 'IceDancing': 7,
 'Kayaking': 8,
 'MilitaryParade': 9,
 'PizzaTossing': 10,
 'PullUps': 11,
 'ShavingBeard': 12,
 'SkateBoarding': 13,
 'SumoWrestling': 14,
 'Surfing': 15,
 'TennisSwing': 16,
 'Typing': 17,
 'WritingOnBoard': 18,
 'YoYo': 19}
```

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

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 20:01:10.435831: 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 instructio
ns in performance-critical operations:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the app
ropriate compiler flags.
2022-08-25 20:01:10.437029: I tensorflow/stream_executor/cuda/cuda_
gpu_executor.cc:975] successful NUMA node read from SysFS had negat
ive value (-1), but there must be at least one NUMA node, so return
ing NUMA node zero
2022-08-25 20:01:10.437301: I tensorflow/stream_executor/cuda/cuda_
gpu_executor.cc:975] successful NUMA node read from SysFS had negat
ive value (-1), but there must be at least one NUMA node, so return
ing NUMA node zero
2022-08-25 20:01:10.437423: I tensorflow/stream_executor/cuda/cuda_
gpu_executor.cc:975] successful NUMA node read from SysFS had negat
ive value (-1), but there must be at least one NUMA node, so return
ing NUMA node zero
2022-08-25 20:01:11.582601: I tensorflow/stream_executor/cuda/cuda_
gpu_executor.cc:975] successful NUMA node read from SysFS had negat
ive value (-1), but there must be at least one NUMA node, so return
ing NUMA node zero
2022-08-25 20:01:11.582860: I tensorflow/stream_executor/cuda/cuda_
gpu_executor.cc:975] successful NUMA node read from SysFS had negat
ive value (-1), but there must be at least one NUMA node, so return
ing NUMA node zero
2022-08-25 20:01:11.583007: I tensorflow/stream_executor/cuda/cuda_
gpu_executor.cc:975] successful NUMA node read from SysFS had negat
ive value (-1), but there must be at least one NUMA node, so return
ing NUMA node zero
2022-08-25 20:01:11.583104: I tensorflow/core/common_runtime/gpu/gp
u_device.cc:1532] Created device /job:localhost/replica:0/task:0/de
vice: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=20,
                       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(20, 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, 20)                660

=================================================================
Total params: 14,732,820
Trainable params: 18,132
Non-trainable params: 14,714,688
_________________________________________________________________
```

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

In [12]:
```
model.save("./models/action-class-20-model-vgg16.h5")
```

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

```
Epoch 1/20

2022-08-25 20:01:16.375293: I tensorflow/stream_executor/cuda/cuda_
dnn.cc:384] Loaded cuDNN version 8401
2022-08-25 20:01:17.769419: I tensorflow/core/platform/default/subp
rocess.cc:304] Start cannot spawn child process: No such file or di
rectory
2022-08-25 20:01:18.062004: W tensorflow/core/common_runtime/bfc_al
locator.cc:290] Allocator (GPU_0_bfc) ran out of memory trying to a
llocate 2.35GiB with freed_by_count=0. The caller indicates that th
is is not a failure, but this may mean that there could be performa
nce gains if more memory were available.

512/512 [==============================] - 93s 172ms/step - loss:
2.3305 - accuracy: 0.3380 - val_loss: 1.6779 - val_accuracy: 0.4874
Epoch 2/20
512/512 [==============================] - 93s 181ms/step - loss:
1.2268 - accuracy: 0.6440 - val_loss: 1.1752 - val_accuracy: 0.6236
Epoch 3/20
512/512 [==============================] - 137s 267ms/step - loss:
0.8810 - accuracy: 0.7327 - val_loss: 1.0025 - val_accuracy: 0.6827
Epoch 4/20
512/512 [==============================] - 148s 289ms/step - loss:
0.7065 - accuracy: 0.7903 - val_loss: 0.9571 - val_accuracy: 0.6992
Epoch 5/20
512/512 [==============================] - 151s 293ms/step - loss:
0.5889 - accuracy: 0.8277 - val_loss: 0.8752 - val_accuracy: 0.7283
Epoch 6/20
512/512 [==============================] - 150s 293ms/step - loss:
0.4970 - accuracy: 0.8537 - val_loss: 0.8522 - val_accuracy: 0.7346
Epoch 7/20
512/512 [==============================] - 152s 297ms/step - loss:
0.4338 - accuracy: 0.8728 - val_loss: 0.8836 - val_accuracy: 0.7205
Epoch 8/20
512/512 [==============================] - 152s 296ms/step - loss:
0.3743 - accuracy: 0.8877 - val_loss: 0.8541 - val_accuracy: 0.7157
Epoch 9/20
512/512 [==============================] - 153s 298ms/step - loss:
0.3270 - accuracy: 0.9000 - val_loss: 0.7805 - val_accuracy: 0.7559
Epoch 10/20
512/512 [==============================] - 152s 296ms/step - loss:
0.2936 - accuracy: 0.9150 - val_loss: 0.7711 - val_accuracy: 0.7638
Epoch 11/20
512/512 [==============================] - 151s 294ms/step - loss:
0.2628 - accuracy: 0.9226 - val_loss: 0.7595 - val_accuracy: 0.7717
Epoch 12/20
512/512 [==============================] - 151s 294ms/step - loss:
0.2314 - accuracy: 0.9345 - val_loss: 0.7738 - val_accuracy: 0.7661
Epoch 13/20
512/512 [==============================] - 152s 296ms/step - loss:
0.2129 - accuracy: 0.9381 - val_loss: 0.7372 - val_accuracy: 0.7748
Epoch 14/20
512/512 [==============================] - 152s 297ms/step - loss:
0.1936 - accuracy: 0.9472 - val_loss: 0.7900 - val_accuracy: 0.7591
Epoch 15/20
512/512 [==============================] - 152s 296ms/step - loss:
0.1729 - accuracy: 0.9529 - val_loss: 0.8902 - val_accuracy: 0.7386
Epoch 16/20
```

```
512/512 [==============================] - 152s 297ms/step - loss:
0.1566 - accuracy: 0.9576 - val_loss: 0.7687 - val_accuracy: 0.7709
Epoch 17/20
512/512 [==============================] - 152s 296ms/step - loss:
0.1402 - accuracy: 0.9625 - val_loss: 0.8218 - val_accuracy: 0.7717
Epoch 18/20
512/512 [==============================] - 152s 296ms/step - loss:
0.1322 - accuracy: 0.9646 - val_loss: 0.8053 - val_accuracy: 0.7677
Epoch 19/20
512/512 [==============================] - 153s 297ms/step - loss:
0.1171 - accuracy: 0.9707 - val_loss: 0.7698 - val_accuracy: 0.7780
Epoch 20/20
512/512 [==============================] - 153s 297ms/step - loss:
0.1068 - accuracy: 0.9721 - val_loss: 0.7963 - val_accuracy: 0.7772
```

In [14]:
```
model.save("./models/action-class-20-trained-vgg16.h5")
```

Evaluate and Predict

In [15]:
```
model = models.load_model("./models/action-class-20-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, 20)                660

=================================================================
Total params: 14,732,820
Trainable params: 18,132
Non-trainable params: 14,714,688
_________________________________________________________________
```

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

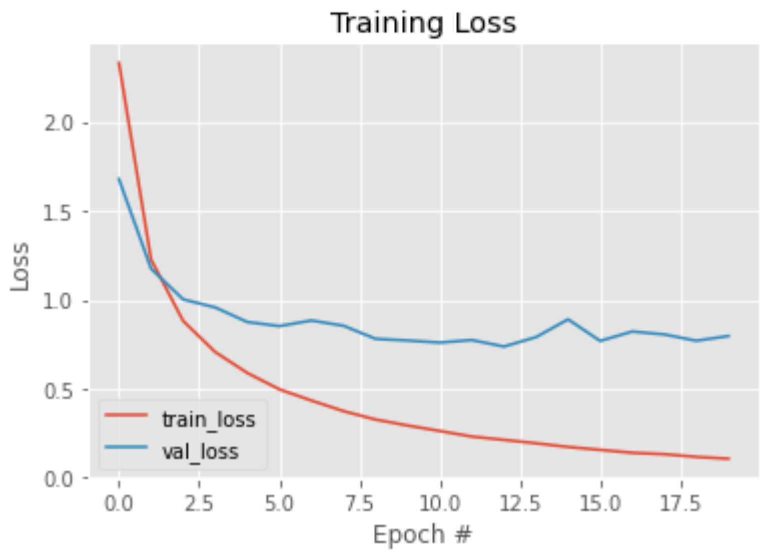
```
127/127 [==============================] - 34s 238ms/step - loss:
0.7963 - accuracy: 0.7772
```

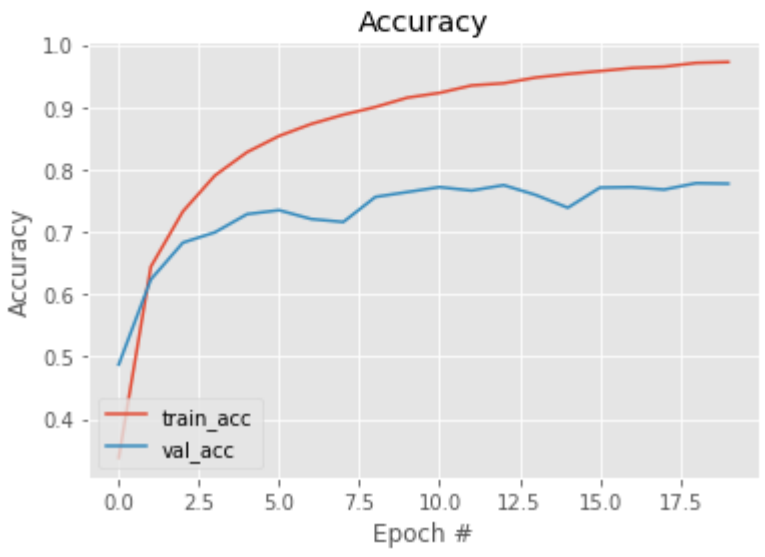
Out[16]:
```
[0.7963224649429321, 0.7771653532981873]
```

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_ac
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: 72.93700769543648
Avg Val Loss: 88.70969265699387
```