In [ ]:

```
import tensorflow as tf
from tensorflow.keras import datasets, models, layers
import matplotlib.pyplot as plt
import numpy as np
from IPython.display import display, Image
```

Functions

In [2]:

```
###Parsing Functions###

#Per TensorFlow documentation;
#Parse the TFRecords for emmbeded features


# Create a dictionary of features
features = {
    'height': tf.io.FixedLenFeature([], tf.int64),
    'width': tf.io.FixedLenFeature([], tf.int64),
    'depth': tf.io.FixedLenFeature([], tf.int64),
    'label': tf.io.FixedLenFeature([], tf.int64),
    'image_raw': tf.io.FixedLenFeature([], tf.string),
}

# Use dictionary to parse data and decode raw image data
def _parse_image_function(example_proto):
    parsed_data = tf.io.parse_single_example(example_proto, features)
    parsed_data['image_raw'] = tf.io.decode_jpeg(parsed_data['image_raw'], channels=3)
    parsed_data['image_raw'] = tf.reshape(parsed_data['image_raw'], [parsed_data['height'], parsed_data['width'], parsed_data['depth']])
    return parsed_data



###PreProcessing###


#Normalize pixel values and return the image and label
def preprocess(features):
    image = tf.cast(features['image_raw'], tf.float32) / 255.0  # normalize to [0,1] range
    return image, features['label']
```

Data

> The data being loaded here are in the recommended TFRecord format created in the TFRecord writer module. The data itself is a collection of jpg images of processed light curves. More information on the lightcurve data process can be found in the pre-data module.

In [3]:

```
#Load TFRecords
train_raw = tf.data.TFRecordDataset('train.tfrecords')
test_raw = tf.data.TFRecordDataset('test.tfrecords')
valid_raw = tf.data.TFRecordDataset('validate.tfrecords')

#Parse the TFRecords
train_parsed = train_raw.map(_parse_image_function)
test_parsed = test_raw.map(_parse_image_function)
valid_parsed = valid_raw.map(_parse_image_function)

for features in train_parsed.take(1):
    print(features['image_raw'].shape)
```

```
(256, 256, 3)
```

In [ ]:

```
#show images in the parsed training file

for features in test_parsed:
    image = features['image_raw']
    label = features['label']
    plt.figure()
    plt.imshow(image)
    plt.title(f'Label: {label}')
    plt.show()
```

In [5]:

```
#Preprocess/normalize data
train_dataset = train_parsed.map(preprocess)
test_dataset = test_parsed.map(preprocess)
valid_dataset = valid_parsed.map(preprocess)
```

In [6]:

```
#Batch and Shuffle

BATCH_SIZE = 32
train_dataset = train_dataset.batch(BATCH_SIZE).prefetch(tf.data.AUTOTUNE)
test_dataset = test_dataset.batch(BATCH_SIZE).prefetch(tf.data.AUTOTUNE)
valid_dataset = valid_dataset.batch(BATCH_SIZE).prefetch(tf.data.AUTOTUNE)
```

In [7]:

```
#Shape confirmation
for images, labels in train_dataset.take(1):
    print(images.shape)
```

```
(32, 256, 256, 3)
```

Model

Feature Extraction

In [8]:

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(256,256, 3))) #Match shape
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
```

Dense Layers

In [9]:

```
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(4))
```

Training

from keras.optimizers import Adam

initial_lr = 0.01 lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(initial_lr, decay_steps=10000, decay_rate=0.9, staircase=True)

In [10]:

```
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

history = model.fit(train_dataset, epochs=5,
                    validation_data=(valid_dataset))
```

```
Epoch 1/5
29/29 [==============================] - 28s 948ms/step - loss: 12.0492 - accuracy: 0.8525 - val_loss: 26.7729 - val_accuracy: 0.4352
Epoch 2/5
29/29 [==============================] - 27s 931ms/step - loss: 5.9998 - accuracy: 0.6042 - val_loss: 29.1460 - val_accuracy: 0.4352
Epoch 3/5
29/29 [==============================] - 27s 926ms/step - loss: 5.4635 - accuracy: 0.7106 - val_loss: 15.4185 - val_accuracy: 0.4352
Epoch 4/5
29/29 [==============================] - 27s 926ms/step - loss: 3.2969 - accuracy: 0.5676 - val_loss: 4.5743 - val_accuracy: 0.4352
Epoch 5/5
29/29 [==============================] - 27s 945ms/step - loss: 1.6142 - accuracy: 0.3902 - val_loss: 1.0097 - val_accuracy: 0.4352
```

In [12]:

```
from keras.optimizers import Adam

initial_lr = 0.01
lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
    initial_lr, decay_steps=10000, decay_rate=0.9, staircase=True
)

optimizer = Adam(learning_rate=lr_schedule)

model.compile(optimizer=optimizer,
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

history = model.fit(train_dataset, epochs=5,
                    validation_data=(valid_dataset))
```

```
Epoch 1/5
29/29 [==============================] - 28s 958ms/step - loss: 860.7210 - accuracy: 0.5333 - val_loss: 0.8461 - val_accuracy: 0.4352
Epoch 2/5
29/29 [==============================] - 27s 934ms/step - loss: 1.8180 - accuracy: 0.6042 - val_loss: 0.8061 - val_accuracy: 0.4352
Epoch 3/5
29/29 [==============================] - 27s 936ms/step - loss: 0.7619 - accuracy: 0.4379 - val_loss: 1.6261 - val_accuracy: 0.4352
Epoch 4/5
29/29 [==============================] - 27s 933ms/step - loss: 1.4274 - accuracy: 0.1364 - val_loss: 0.9896 - val_accuracy: 0.5648
Epoch 5/5
29/29 [==============================] - 27s 930ms/step - loss: 0.9021 - accuracy: 0.5621 - val_loss: 0.8009 - val_accuracy: 0.5648
```

Testing

In [13]:

```
test_loss, test_acc = model.evaluate(test_dataset, verbose=2)
print(test_acc)
```

```
7/7 - 1s - loss: 0.8009 - accuracy: 0.5648 - 1s/epoch - 181ms/step
0.5647668242454529
```

In [ ]: