```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly
remount, call drive.mount("/content/drive", force_remount=True).
```

```
%cd /content/drive/My Drive/
```

```
/content/drive/My Drive
```

```
import numpy as np
import tensorflow as tf
from tensorflow import keras
import pandas as pd
from sklearn.model_selection import train_test_split
import urllib.request

urllib.request.urlretrieve("https://zenodo.org/record/3164691/files/
QG_jets_10.npz", "QG_jets.npz")
```

```
('QG_jets.npz', <http.client.HTTPMessage at 0x7fa45a4ac550>)
```

Task II: Classical Graph Neural Network (GNN) Part

Use ParticleNet's data for Quark/Gluon jet classification. Choose 2 Graph-based architectures of your choice to classify jets as being quarks or gluons. Provide a description on what considerations you have taken to project this point-cloud dataset to a set of interconnected nodes and edges. Discuss the resulting performance of the 2 chosen architectures.

Overview of the Dataset

The input file has 100k jets, exactly 50k quark and 50k gluon jets, randomly sorted. Every jet in the file contains M x F data where M is the maximum number of multiplicity of the jets in the file and F is the number of each particle's features (pt, rapidity, azimuthal angle, and Particle Data Group ID).

Graph-Based Architectures for Jet Classification: Selection and Considerations

For this task, we will be using the Jet Classifciation paper model of Particle-Net Lite and its code for jet classification.

```
df = np.load('QG_jets.npz')
lst = df.files
print(list(df.keys()))
```

```
['X', 'y']
```

To enable the use of a softmax activation function used at the end of the ParticleNet-Lite networks, which is necessary for classification, the labels must be transformed into a one-hot format. This ensures that the output of the network represents the probability of the input particle belonging to each class, with a total probability of 1.0.

```
# Extract data as per keys
x = df['X']
y = df['y']

# Transform the labels to one-hot format
y = tf.keras.utils.to_categorical(y)

num_samples = x.shape[0]
num_particles = x.shape[1]
num_features = x.shape[2]
print("Total number of samples:", num_samples)
print("Maximum number of particles in a jet:", num_particles)
print("Number of features:", num_features)
```

```
Total number of samples: 100000
Maximum number of particles in a jet: 138
Number of features: 4
```

Preprocessing

In the paper, dataset consisting of 2 million jets is split into 1.6M/200k/200k for training, validation and testing. So we follow similar 80/10/10 splitting.

```
x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.2, random_state=42)
x_test, x_val, y_test, y_val = train_test_split(x_test, y_test,
test_size=0.5, random_state=21)

print(x_train.shape, y_train.shape)
print(x_test.shape, y_test.shape)
print(x_val.shape, y_val.shape)
```

```
(80000, 138, 4) (80000, 2)
(10000, 138, 4) (10000, 2)
(10000, 138, 4) (10000, 2)
```

The model takes three input arrays:

points: the coordinates of the particles in the (eta, phi) space. It should be an array with a shape of (N, P, 2), where N is the batch size and P is the number of particles.

features: the features of the particles. It should be an array with a shape of (N, P, C), where N is the batch size, C is the number of features, and P is the number of particles.

mask: a mask array with a shape of (N, P, 1), taking a value of 0 for padded positions.

```
train_set = {
    # the coordinates of the particles in the (eta, phi) space
    'points': x_train[:, :, 1:3],
```

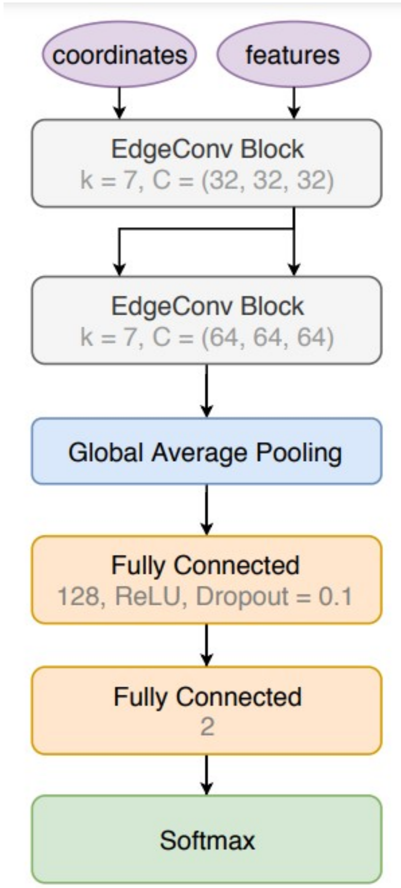
```
    # the features of the particles
    'features': x_train,
    # a mask array, taking a value of 0 for padded positions
    'mask': np.array(np.sum(x_train, axis=2) != 0,
np.float32).reshape(len(x_train), num_particles, 1)
}

val_set = {
    'points': x_val[:, :, 1:3],
    'features': x_val,
    'mask': np.array(np.sum(x_val, axis=2) != 0,
np.float32).reshape(len(x_val), num_particles, 1)
}

test_set = {
    'points': x_test[:, :, 1:3],
    'features': x_test,
    'mask': np.array(np.sum(x_test, axis=2) != 0,
np.float32).reshape(len(x_test), num_particles, 1)
}

input_shape = {k:train_set[k].shape[1:] for k in train_set}
input_shape
```

```
{'points': (138, 2), 'features': (138, 4), 'mask': (138, 1)}
```

ParticleNet-Lite Architecture

About the architecture

The ParticleNet-Lite architecture used here consists of two EdgeConv blocks. The first EdgeConv block uses the spatial coordinates of the particles in the pseudorapidity-azimuth space to compute the distances, while the subsequent blocks use the learned feature vectors as coordinates. The number of nearest neighbors k is 7 for all three blocks, and the number of channels C for each EdgeConv block is (32, 32, 32) and (64, 64, 64) respectively. After the EdgeConv blocks, a channel-wise global average pooling operation is applied to aggregate the learned features over all particles in the cloud. This is followed by a fully connected layer with 128 units and the ReLU activation. A dropout layer with a drop probability of 0.1 is included to prevent overfitting. A fully connected layer with two units, followed by a softmax function, is used to generate the output for the binary classification.



Implementation

```
import tf_keras_model
from tf_keras_model import get_particle_net_lite

num_classes = 2
model = get_particle_net_lite(num_classes, input_shape)

import logging
logging.basicConfig(level=logging.INFO,
                    format='[%(asctime)s] %(levelname)s: %(message)s')
# Training parameters
batch_size = 1024
epochs = 25
def lr_schedule(epoch):
    lr = 1e-3
    if epoch > 10:
        lr *= 0.1
    elif epoch > 20:
        lr *= 0.01
    logging.info('Learning rate: %f' % lr)
    return lr
model.compile(loss='categorical_crossentropy',
```

```
optimizer=keras.optimizers.Adam(learning_rate=lr_schedule(0)),
              metrics=['accuracy'])
model.summary()
```

```
Model: "ParticleNet"
________________________________________________________________________________
____________________________
 Layer (type)                   Output Shape         Param #
Connected to
================================================================================
============================
 mask (InputLayer)              [(None, 138, 1)]     0            []



 tf.math.not_equal (TFOpLambda)  (None, 138, 1)      0
['mask[0][0]']


 tf.cast (TFOpLambda)           (None, 138, 1)       0
['tf.math.not_equal[0][0]']


 tf.math.equal (TFOpLambda)     (None, 138, 1)       0
['tf.cast[0][0]']


 tf.cast_1 (TFOpLambda)         (None, 138, 1)       0
['tf.math.equal[0][0]']


 tf.math.multiply (TFOpLambda)  (None, 138, 1)       0
['tf.cast_1[0][0]']


 points (InputLayer)            [(None, 138, 2)]     0            []



 tf.math.add (TFOpLambda)       (None, 138, 2)       0
['tf.math.multiply[0][0]',

'points[0][0]']


 features (InputLayer)          [(None, 138, 4)]     0            []
```

```
 tf.compat.v1.transpose (TFOpLa  (None, 2, 138)      0
['tf.math.add[0][0]']
 mbda)


 tf.expand_dims (TFOpLambda)    (None, 138, 1, 4)    0
['features[0][0]']


 tf.math.multiply_1 (TFOpLambda  (None, 138, 2)      0
['tf.math.add[0][0]',
 )
'tf.math.add[0][0]']


 tf.linalg.matmul (TFOpLambda)  (None, 138, 138)     0
['tf.math.add[0][0]',

'tf.compat.v1.transpose[0][0]']


 tf.math.multiply_2 (TFOpLambda  (None, 138, 2)      0
['tf.math.add[0][0]',
 )
'tf.math.add[0][0]']


 ParticleNet_fts_bn (BatchNorma  (None, 138, 1, 4)   16
['tf.expand_dims[0][0]']
 lization)


 tf.math.reduce_sum (TFOpLambda  (None, 138, 1)      0
['tf.math.multiply_1[0][0]']
 )


 tf.math.multiply_3 (TFOpLambda  (None, 138, 138)    0
['tf.linalg.matmul[0][0]']
 )


 tf.math.reduce_sum_1 (TFOpLamb  (None, 138, 1)      0
['tf.math.multiply_2[0][0]']
 da)
```

```
 tf.compat.v1.squeeze (TFOpLamb  (None, 138, 4)      0
['ParticleNet_fts_bn[0][0]']
 da)



 tf.math.subtract (TFOpLambda)  (None, 138, 138)     0
['tf.math.reduce_sum[0][0]',

'tf.math.multiply_3[0][0]']


 tf.compat.v1.transpose_1 (TFOp  (None, 1, 138)      0
['tf.math.reduce_sum_1[0][0]']
 Lambda)



 tf.compat.v1.shape (TFOpLambda  (3,)                0
['tf.compat.v1.squeeze[0][0]']
 )



 tf.__operators__.add (TFOpLamb  (None, 138, 138)    0
['tf.math.subtract[0][0]',
 da)
'tf.compat.v1.transpose_1[0][0]'
                                                                   ]



 tf.__operators__.getitem_1 (Sl  ()                  0
['tf.compat.v1.shape[0][0]']
 icingOpLambda)



 tf.math.negative (TFOpLambda)  (None, 138, 138)     0
['tf.__operators__.add[0][0]']


 tf.range (TFOpLambda)          (None,)              0
['tf.__operators__.getitem_1[0][0
                                                                   ]']
```

```
 tf.math.top_k (TFOpLambda)     TopKV2(values=(None  0
['tf.math.negative[0][0]']
                                , 138, 8),

                                 indices=(None, 138

                                , 8))



 tf.reshape (TFOpLambda)        (None, 1, 1, 1)      0
['tf.range[0][0]']


 tf.__operators__.getitem (Slic  (None, 138, 7)      0
['tf.math.top_k[0][1]']
 ingOpLambda)



 tf.tile (TFOpLambda)           (None, 138, 7, 1)    0
['tf.reshape[0][0]']


 tf.expand_dims_1 (TFOpLambda)  (None, 138, 7, 1)    0
['tf.__operators__.getitem[0][0]'
                                                                 ]



 tf.expand_dims_2 (TFOpLambda)  (None, 138, 1, 4)    0
['tf.compat.v1.squeeze[0][0]']


 tf.concat (TFOpLambda)         (None, 138, 7, 2)    0
['tf.tile[0][0]',

'tf.expand_dims_1[0][0]']


 tf.tile_1 (TFOpLambda)         (None, 138, 7, 4)    0
['tf.expand_dims_2[0][0]']


 tf.compat.v1.gather_nd (TFOpLa  (None, 138, 7, 4)   0
['tf.compat.v1.squeeze[0][0]',
 mbda)
```

```
'tf.concat[0][0]']


 tf.math.subtract_1 (TFOpLambda  (None, 138, 7, 4)   0
['tf.compat.v1.gather_nd[0][0]',
 )
'tf.tile_1[0][0]']


 tf.concat_1 (TFOpLambda)       (None, 138, 7, 8)    0
['tf.tile_1[0][0]',

'tf.math.subtract_1[0][0]']


 ParticleNet_EdgeConv0_conv0 (C  (None, 138, 7, 32)  256
['tf.concat_1[0][0]']
 onv2D)



 ParticleNet_EdgeConv0_bn0 (Bat  (None, 138, 7, 32)  128
['ParticleNet_EdgeConv0_conv0[0][
 chNormalization)                                                0]']



 ParticleNet_EdgeConv0_act0 (Ac  (None, 138, 7, 32)  0
['ParticleNet_EdgeConv0_bn0[0][0]
 tivation)                                                       ']



 ParticleNet_EdgeConv0_conv1 (C  (None, 138, 7, 32)  1024
['ParticleNet_EdgeConv0_act0[0][0
 onv2D)                                                          ]']



 ParticleNet_EdgeConv0_bn1 (Bat  (None, 138, 7, 32)  128
['ParticleNet_EdgeConv0_conv1[0][
 chNormalization)                                                0]']



 ParticleNet_EdgeConv0_act1 (Ac  (None, 138, 7, 32)  0
['ParticleNet_EdgeConv0_bn1[0][0]
 tivation)                                                       ']
```

```
 tf.expand_dims_3 (TFOpLambda)  (None, 138, 1, 4)    0
['tf.compat.v1.squeeze[0][0]']


 ParticleNet_EdgeConv0_conv2 (C  (None, 138, 7, 32)  1024
['ParticleNet_EdgeConv0_act1[0][0
 onv2D)                                                          ]']



 ParticleNet_EdgeConv0_sc_conv   (None, 138, 1, 32)  128
['tf.expand_dims_3[0][0]']
 (Conv2D)



 ParticleNet_EdgeConv0_bn2 (Bat  (None, 138, 7, 32)  128
['ParticleNet_EdgeConv0_conv2[0][
 chNormalization)                                                0]']



 ParticleNet_EdgeConv0_sc_bn (B  (None, 138, 1, 32)  128
['ParticleNet_EdgeConv0_sc_conv[0
 atchNormalization)                                              ]
[0]']


 ParticleNet_EdgeConv0_act2 (Ac  (None, 138, 7, 32)  0
['ParticleNet_EdgeConv0_bn2[0][0]
 tivation)                                                       ']



 tf.compat.v1.squeeze_1 (TFOpLa  (None, 138, 32)     0
['ParticleNet_EdgeConv0_sc_bn[0][
 mbda)                                                           0]']



 tf.math.reduce_mean (TFOpLambd  (None, 138, 32)     0
['ParticleNet_EdgeConv0_act2[0][0
 a)                                                              ]']



 tf.__operators__.add_1 (TFOpLa  (None, 138, 32)     0
```

```
['tf.compat.v1.squeeze_1[0][0]',
 mbda)
'tf.math.reduce_mean[0][0]']


 ParticleNet_EdgeConv0_sc_act (  (None, 138, 32)     0
['tf.__operators__.add_1[0][0]']
 Activation)



 tf.math.add_1 (TFOpLambda)     (None, 138, 32)      0
['tf.math.multiply[0][0]',

'ParticleNet_EdgeConv0_sc_act[0]
                                                                    [0]']



 tf.compat.v1.transpose_2 (TFOp  (None, 32, 138)     0
['tf.math.add_1[0][0]']
 Lambda)



 tf.math.multiply_4 (TFOpLambda  (None, 138, 32)     0
['tf.math.add_1[0][0]',
 )
'tf.math.add_1[0][0]']


 tf.linalg.matmul_1 (TFOpLambda  (None, 138, 138)    0
['tf.math.add_1[0][0]',
 )
'tf.compat.v1.transpose_2[0][0]'
                                                                    ]



 tf.math.multiply_5 (TFOpLambda  (None, 138, 32)     0
['tf.math.add_1[0][0]',
 )
'tf.math.add_1[0][0]']


 tf.math.reduce_sum_2 (TFOpLamb  (None, 138, 1)      0
['tf.math.multiply_4[0][0]']
 da)
```

```
 tf.math.multiply_6 (TFOpLambda  (None, 138, 138)    0
['tf.linalg.matmul_1[0][0]']
 )


 tf.math.reduce_sum_3 (TFOpLamb  (None, 138, 1)      0
['tf.math.multiply_5[0][0]']
 da)


 tf.math.subtract_2 (TFOpLambda  (None, 138, 138)    0
['tf.math.reduce_sum_2[0][0]',
 )
'tf.math.multiply_6[0][0]']

 tf.compat.v1.transpose_3 (TFOp  (None, 1, 138)      0
['tf.math.reduce_sum_3[0][0]']
 Lambda)


 tf.compat.v1.shape_1 (TFOpLamb  (3,)                0
['ParticleNet_EdgeConv0_sc_act[0]
 da)                                                       [0]']


 tf.__operators__.add_2 (TFOpLa  (None, 138, 138)    0
['tf.math.subtract_2[0][0]',
 mbda)
'tf.compat.v1.transpose_3[0][0]'
                                                           ]


 tf.__operators__.getitem_3 (Sl  ()                  0
['tf.compat.v1.shape_1[0][0]']
 icingOpLambda)


 tf.math.negative_1 (TFOpLambda  (None, 138, 138)    0
['tf.__operators__.add_2[0][0]']
 )
```

```
 tf.range_1 (TFOpLambda)        (None,)              0
['tf.__operators__.getitem_3[0][0
                                                                ]']

 tf.math.top_k_1 (TFOpLambda)   TopKV2(values=(None  0
['tf.math.negative_1[0][0]']
                                , 138, 8),

                                 indices=(None, 138

                                , 8))

 tf.reshape_1 (TFOpLambda)      (None, 1, 1, 1)      0
['tf.range_1[0][0]']

 tf.__operators__.getitem_2 (Sl  (None, 138, 7)      0
['tf.math.top_k_1[0][1]']
 icingOpLambda)

 tf.tile_2 (TFOpLambda)         (None, 138, 7, 1)    0
['tf.reshape_1[0][0]']

 tf.expand_dims_4 (TFOpLambda)  (None, 138, 7, 1)    0
['tf.__operators__.getitem_2[0][0
                                                                ]']

 tf.expand_dims_5 (TFOpLambda)  (None, 138, 1, 32)   0
['ParticleNet_EdgeConv0_sc_act[0]
                                                                [0]']

 tf.concat_2 (TFOpLambda)       (None, 138, 7, 2)    0
['tf.tile_2[0][0]',
```

```
'tf.expand_dims_4[0][0]']


 tf.tile_3 (TFOpLambda)         (None, 138, 7, 32)   0
['tf.expand_dims_5[0][0]']


 tf.compat.v1.gather_nd_1 (TFOp  (None, 138, 7, 32)  0
['ParticleNet_EdgeConv0_sc_act[0]
 Lambda)                                                           [0]',


'tf.concat_2[0][0]']


 tf.math.subtract_3 (TFOpLambda  (None, 138, 7, 32)  0
['tf.compat.v1.gather_nd_1[0][0]'
 )                                                                 ,
'tf.tile_3[0][0]']


 tf.concat_3 (TFOpLambda)       (None, 138, 7, 64)   0
['tf.tile_3[0][0]',

'tf.math.subtract_3[0][0]']


 ParticleNet_EdgeConv1_conv0 (C  (None, 138, 7, 64)  4096
['tf.concat_3[0][0]']
 onv2D)



 ParticleNet_EdgeConv1_bn0 (Bat  (None, 138, 7, 64)  256
['ParticleNet_EdgeConv1_conv0[0][
 chNormalization)                                                  0]']



 ParticleNet_EdgeConv1_act0 (Ac  (None, 138, 7, 64)  0
['ParticleNet_EdgeConv1_bn0[0][0]
 tivation)                                                         ']



 ParticleNet_EdgeConv1_conv1 (C  (None, 138, 7, 64)  4096
['ParticleNet_EdgeConv1_act0[0][0
 onv2D)                                                            ]']
```

```
 ParticleNet_EdgeConv1_bn1 (Bat  (None, 138, 7, 64)  256
['ParticleNet_EdgeConv1_conv1[0][
 chNormalization)                                                0]']


 ParticleNet_EdgeConv1_act1 (Ac  (None, 138, 7, 64)  0
['ParticleNet_EdgeConv1_bn1[0][0]
 tivation)                                                       ']


 tf.expand_dims_6 (TFOpLambda)  (None, 138, 1, 32)   0
['ParticleNet_EdgeConv0_sc_act[0]
                                                                 [0]']


 ParticleNet_EdgeConv1_conv2 (C  (None, 138, 7, 64)  4096
['ParticleNet_EdgeConv1_act1[0][0
 onv2D)                                                          ]']


 ParticleNet_EdgeConv1_sc_conv   (None, 138, 1, 64)  2048
['tf.expand_dims_6[0][0]']
 (Conv2D)


 ParticleNet_EdgeConv1_bn2 (Bat  (None, 138, 7, 64)  256
['ParticleNet_EdgeConv1_conv2[0][
 chNormalization)                                                0]']


 ParticleNet_EdgeConv1_sc_bn (B  (None, 138, 1, 64)  256
['ParticleNet_EdgeConv1_sc_conv[0
 atchNormalization)                                              ]
[0]']

 ParticleNet_EdgeConv1_act2 (Ac  (None, 138, 7, 64)  0
['ParticleNet_EdgeConv1_bn2[0][0]
 tivation)                                                       ']
```

```
 tf.compat.v1.squeeze_2 (TFOpLa  (None, 138, 64)     0
['ParticleNet_EdgeConv1_sc_bn[0][
 mbda)                                                           0]']


 tf.math.reduce_mean_1 (TFOpLam  (None, 138, 64)     0
['ParticleNet_EdgeConv1_act2[0][0
 bda)                                                            ]']


 tf.__operators__.add_3 (TFOpLa  (None, 138, 64)     0
['tf.compat.v1.squeeze_2[0][0]',
 mbda)
'tf.math.reduce_mean_1[0][0]']


 ParticleNet_EdgeConv1_sc_act (  (None, 138, 64)     0
['tf.__operators__.add_3[0][0]']
 Activation)


 tf.math.multiply_7 (TFOpLambda  (None, 138, 64)     0
['ParticleNet_EdgeConv1_sc_act[0]
 )                                                               [0]',

'tf.cast[0][0]']


 tf.math.reduce_mean_2 (TFOpLam  (None, 64)          0
['tf.math.multiply_7[0][0]']
 bda)


 dense (Dense)                  (None, 128)          8320
['tf.math.reduce_mean_2[0][0]']


 dropout (Dropout)              (None, 128)          0
['dense[0][0]']


 dense_1 (Dense)                (None, 2)            258
['dropout[0][0]']
```

```
==========================================================================
=============================
Total params: 26,898
Trainable params: 26,122
Non-trainable params: 776
__________________________________________________________________________
_____________________________

import os
save_dir = 'model_checkpoints'
model_name = '%s_model.{epoch:03d}.h5' % 'particle_net_lite'
if not os.path.isdir(save_dir):
    os.makedirs(save_dir)
filepath = os.path.join(save_dir, model_name)

# Prepare callbacks for model saving and for learning rate adjustment.
checkpoint = keras.callbacks.ModelCheckpoint(filepath=filepath,
                                             monitor='val_accuracy',
                                             verbose=1,
                                             save_best_only=True)

lr_scheduler = keras.callbacks.LearningRateScheduler(lr_schedule)
progress_bar = keras.callbacks.ProgbarLogger()
callbacks = [checkpoint, lr_scheduler, progress_bar]
history = model.fit(train_set, y_train,
                    batch_size=batch_size,
                    epochs=epochs,
                    validation_data=(val_set, y_val),
                    shuffle=True,
                    callbacks=callbacks)

Epoch 1/100
      0/Unknown - 490s 0s/sample - loss: 0.5310 - accuracy: 0.7362
Epoch 1: val_accuracy improved from -inf to 0.60640, saving model to
model_checkpoints/particle_net_lite_model.001.h5
79/79 [==============================] - 504s 6s/sample - loss: 0.5310
- accuracy: 0.7362 - val_loss: 0.6167 - val_accuracy: 0.6064 - lr:
0.0010
Epoch 2/100
 0/79 [..............................] - ETA: 0s - loss: 0.4794 -
accuracy: 0.7765
Epoch 2: val_accuracy improved from 0.60640 to 0.74290, saving model
to model_checkpoints/particle_net_lite_model.002.h5
79/79 [==============================] - 496s 6s/sample - loss: 0.4794
- accuracy: 0.7765 - val_loss: 0.5353 - val_accuracy: 0.7429 - lr:
0.0010
Epoch 3/100
 0/79 [..............................] - ETA: 0s - loss: 0.4711 -
```

```
accuracy: 0.7845
Epoch 3: val_accuracy improved from 0.74290 to 0.77000, saving model
to model_checkpoints/particle_net_lite_model.003.h5
79/79 [==============================] - 490s 6s/sample - loss: 0.4711
- accuracy: 0.7845 - val_loss: 0.5035 - val_accuracy: 0.7700 - lr:
0.0010
Epoch 4/100
 0/79 [..............................] - ETA: 0s - loss: 0.4645 -
accuracy: 0.7885
Epoch 4: val_accuracy improved from 0.77000 to 0.78950, saving model
to model_checkpoints/particle_net_lite_model.004.h5
79/79 [==============================] - 478s 6s/sample - loss: 0.4645
- accuracy: 0.7885 - val_loss: 0.4778 - val_accuracy: 0.7895 - lr:
0.0010
Epoch 5/100
 0/79 [..............................] - ETA: 0s - loss: 0.4585 -
accuracy: 0.7929
Epoch 5: val_accuracy did not improve from 0.78950
79/79 [==============================] - 468s 6s/sample - loss: 0.4585
- accuracy: 0.7929 - val_loss: 0.4651 - val_accuracy: 0.7868 - lr:
0.0010
Epoch 6/100
 0/79 [..............................] - ETA: 0s - loss: 0.4533 -
accuracy: 0.7958
Epoch 6: val_accuracy improved from 0.78950 to 0.79210, saving model
to model_checkpoints/particle_net_lite_model.006.h5
79/79 [==============================] - 472s 6s/sample - loss: 0.4533
- accuracy: 0.7958 - val_loss: 0.4598 - val_accuracy: 0.7921 - lr:
0.0010
Epoch 7/100
 0/79 [..............................] - ETA: 0s - loss: 0.4510 -
accuracy: 0.7973
Epoch 7: val_accuracy improved from 0.79210 to 0.79870, saving model
to model_checkpoints/particle_net_lite_model.007.h5
79/79 [==============================] - 472s 6s/sample - loss: 0.4510
- accuracy: 0.7973 - val_loss: 0.4525 - val_accuracy: 0.7987 - lr:
0.0010
Epoch 8/100
 0/79 [..............................] - ETA: 0s - loss: 0.4462 -
accuracy: 0.8004
Epoch 8: val_accuracy did not improve from 0.79870
79/79 [==============================] - 477s 6s/sample - loss: 0.4462
- accuracy: 0.8004 - val_loss: 0.4598 - val_accuracy: 0.7958 - lr:
0.0010
Epoch 9/100
 0/79 [..............................] - ETA: 0s - loss: 0.4438 -
accuracy: 0.8018
Epoch 9: val_accuracy did not improve from 0.79870
79/79 [==============================] - 468s 6s/sample - loss: 0.4438
- accuracy: 0.8018 - val_loss: 0.4604 - val_accuracy: 0.7920 - lr:
```

```
0.0010
Epoch 10/100
 0/79 [..............................] - ETA: 0s - loss: 0.4405 -
accuracy: 0.8034
Epoch 10: val_accuracy did not improve from 0.79870
79/79 [==============================] - 470s 6s/sample - loss: 0.4405
- accuracy: 0.8034 - val_loss: 0.4649 - val_accuracy: 0.7841 - lr:
0.0010
Epoch 11/100
 0/79 [..............................] - ETA: 0s - loss: 0.4404 -
accuracy: 0.8031
Epoch 11: val_accuracy improved from 0.79870 to 0.80290, saving model
to model_checkpoints/particle_net_lite_model.011.h5
79/79 [==============================] - 470s 6s/sample - loss: 0.4404
- accuracy: 0.8031 - val_loss: 0.4442 - val_accuracy: 0.8029 - lr:
0.0010
Epoch 12/100
 0/79 [..............................] - ETA: 0s - loss: 0.4348 -
accuracy: 0.8071
Epoch 12: val_accuracy improved from 0.80290 to 0.81010, saving model
to model_checkpoints/particle_net_lite_model.012.h5
79/79 [==============================] - 464s 6s/sample - loss: 0.4348
- accuracy: 0.8071 - val_loss: 0.4381 - val_accuracy: 0.8101 - lr:
1.0000e-04
Epoch 13/100
 0/79 [..............................] - ETA: 0s - loss: 0.4327 -
accuracy: 0.8074
Epoch 13: val_accuracy did not improve from 0.81010
79/79 [==============================] - 464s 6s/sample - loss: 0.4327
- accuracy: 0.8074 - val_loss: 0.4377 - val_accuracy: 0.8076 - lr:
1.0000e-04
Epoch 14/100
 0/79 [..............................] - ETA: 0s - loss: 0.4320 -
accuracy: 0.8080
Epoch 14: val_accuracy did not improve from 0.81010
79/79 [==============================] - 465s 6s/sample - loss: 0.4320
- accuracy: 0.8080 - val_loss: 0.4368 - val_accuracy: 0.8093 - lr:
1.0000e-04
Epoch 15/100
 0/79 [..............................] - ETA: 0s - loss: 0.4319 -
accuracy: 0.8077
Epoch 15: val_accuracy did not improve from 0.81010
79/79 [==============================] - 469s 6s/sample - loss: 0.4319
- accuracy: 0.8077 - val_loss: 0.4367 - val_accuracy: 0.8092 - lr:
1.0000e-04
Epoch 16/100
 0/79 [..............................] - ETA: 0s - loss: 0.4314 -
accuracy: 0.8084
Epoch 16: val_accuracy did not improve from 0.81010
79/79 [==============================] - 464s 6s/sample - loss: 0.4314
```

```
- accuracy: 0.8084 - val_loss: 0.4360 - val_accuracy: 0.8099 - lr:
1.0000e-04
Epoch 17/100
 0/79 [..............................] - ETA: 0s - loss: 0.4310 -
accuracy: 0.8090
Epoch 17: val_accuracy did not improve from 0.81010
79/79 [==============================] - 512s 6s/sample - loss: 0.4310
- accuracy: 0.8090 - val_loss: 0.4363 - val_accuracy: 0.8100 - lr:
1.0000e-04
Epoch 18/100
 0/79 [..............................] - ETA: 0s - loss: 0.4308 -
accuracy: 0.8091
Epoch 18: val_accuracy did not improve from 0.81010
79/79 [==============================] - 487s 6s/sample - loss: 0.4308
- accuracy: 0.8091 - val_loss: 0.4361 - val_accuracy: 0.8101 - lr:
1.0000e-04
Epoch 19/100
 0/79 [..............................] - ETA: 0s - loss: 0.4305 -
accuracy: 0.8084
Epoch 19: val_accuracy improved from 0.81010 to 0.81190, saving model
to model_checkpoints/particle_net_lite_model.019.h5
79/79 [==============================] - 501s 6s/sample - loss: 0.4305
- accuracy: 0.8084 - val_loss: 0.4359 - val_accuracy: 0.8119 - lr:
1.0000e-04
Epoch 20/100
 0/79 [..............................] - ETA: 0s - loss: 0.4302 -
accuracy: 0.8097
Epoch 20: val_accuracy did not improve from 0.81190
79/79 [==============================] - 473s 6s/sample - loss: 0.4302
- accuracy: 0.8097 - val_loss: 0.4381 - val_accuracy: 0.8057 - lr:
1.0000e-04
Epoch 21/100
 0/79 [..............................] - ETA: 0s - loss: 0.4300 -
accuracy: 0.8088
Epoch 21: val_accuracy did not improve from 0.81190
79/79 [==============================] - 484s 6s/sample - loss: 0.4300
- accuracy: 0.8088 - val_loss: 0.4355 - val_accuracy: 0.8106 - lr:
1.0000e-04
Epoch 22/100
 0/79 [..............................] - ETA: 0s - loss: 0.4301 -
accuracy: 0.8091
Epoch 22: val_accuracy did not improve from 0.81190
79/79 [==============================] - 488s 6s/sample - loss: 0.4301
- accuracy: 0.8091 - val_loss: 0.4352 - val_accuracy: 0.8088 - lr:
1.0000e-04
Epoch 23/100
 0/79 [..............................] - ETA: 0s - loss: 0.4291 -
accuracy: 0.8097
Epoch 23: val_accuracy improved from 0.81190 to 0.81280, saving model
to model_checkpoints/particle_net_lite_model.023.h5
```

```
79/79 [==============================] - 476s 6s/sample - loss: 0.4291
- accuracy: 0.8097 - val_loss: 0.4355 - val_accuracy: 0.8128 - lr:
1.0000e-04
Epoch 24/100
 0/79 [..............................] - ETA: 0s - loss: 0.4292 -
accuracy: 0.8096
Epoch 24: val_accuracy did not improve from 0.81280
79/79 [==============================] - 474s 6s/sample - loss: 0.4292
- accuracy: 0.8096 - val_loss: 0.4348 - val_accuracy: 0.8095 - lr:
1.0000e-04
Epoch 25/100
 0/79 [..............................] - ETA: 0s - loss: 0.4286 -
accuracy: 0.8103
Epoch 25: val_accuracy did not improve from 0.81280
79/79 [==============================] - 480s 6s/sample - loss: 0.4286
- accuracy: 0.8103 - val_loss: 0.4358 - val_accuracy: 0.8120 - lr:
1.0000e-04
```

Evaluation

```
import json
import matplotlib.pyplot as plt
with open("particle_net_lite_history.json", "w") as outfile:
    outfile.write(str(history.history))

result = model.evaluate(test_dataset, y_test)
print("test loss, test acc:", result)

test loss, test acc: [0.5085463461276892, 0.7416348924512864]
```

Code Summary

We used the ParticleNet-Lite model for Jet Classification. The training loss decreases over epochs and also the training accuracy increases which means the model fits the data. However, the validation loss and accuracy becomes nearly constant after near 20th epoch which shows the overfitting. The training accuracy is near 81% which is comparable to the results of the paper Jet tagging via particle clouds.