Pointnet\_expanded\_our1:

def tnet(inputs, num\_features):

# Initalise bias as the indentity matrix

bias = keras.initializers.Constant(np.eye(num\_features).flatten())

reg = OrthogonalRegularizer(num\_features)

x = conv\_bn(inputs, 32)

x = conv\_bn(x, 128)

x = conv\_bn(x, 256)

x = conv\_bn(x, 1028)

x = layers.GlobalMaxPooling1D()(x)

x = dense\_bn(x, 512)

x = dense\_bn(x, 256)

x = dense\_bn(x, 128)

x = layers.Dense(

num\_features \* num\_features,

kernel\_initializer="zeros",

bias\_initializer=bias,

activity\_regularizer=reg,

)(x)

feat\_T = layers.Reshape((num\_features, num\_features))(x)

# Apply affine transformation to input features

return layers.Dot(axes=(2, 1))([inputs, feat\_T])

x = tnet(inputs, 3)

x = conv\_bn(x, 32)

x = conv\_bn(x, 32)

x = tnet(x, 32)

x = conv\_bn(x, 32)

x = conv\_bn(x, 64)

x = conv\_bn(x, 128)

x = conv\_bn(x, 512)

x = layers.GlobalMaxPooling1D()(x)

x = dense\_bn(x, 256)

x = layers.Dropout(0.3)(x)

x = dense\_bn(x, 128)

x = layers.Dropout(0.3)(x)

Pointnet\_expanded\_our2:

def tnet(inputs, num\_features):

# Initalise bias as the indentity matrix

bias = keras.initializers.Constant(np.eye(num\_features).flatten())

reg = OrthogonalRegularizer(num\_features)

x = conv\_bn(inputs, 32)

x = conv\_bn(x, 128)

x = conv\_bn(x, 256)

x = conv\_bn(x, 1028)

x = layers.GlobalMaxPooling1D()(x)

x = dense\_bn(x, 512)

x = dense\_bn(x, 256)

x = dense\_bn(x, 128)

x = layers.Dense(

num\_features \* num\_features,

kernel\_initializer="zeros",

bias\_initializer=bias,

activity\_regularizer=reg,

)(x)

feat\_T = layers.Reshape((num\_features, num\_features))(x)

# Apply affine transformation to input features

return layers.Dot(axes=(2, 1))([inputs, feat\_T])

x = tnet(inputs, 3)

x = conv\_bn(x, 32)

x = conv\_bn(x, 32)

x = tnet(x, 32)

x = conv\_bn(x, 32)

x = conv\_bn(x, 64)

x = conv\_bn(x, 128)

x = conv\_bn(x, 512)

x = layers.GlobalMaxPooling1D()(x)

x = dense\_bn(x, 512)

x = dense\_bn(x, 256)

x = layers.Dropout(0.3)(x)

x = dense\_bn(x, 128)

x = layers.Dropout(0.3)(x)