In [1]:

```
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as pt

from tensorflow.keras import layers
from tensorflow.keras import activations as act
from tensorflow.keras.datasets.cifar10 import load_data
```

In [2]:

```
(x_train, y_train), (x_test, y_test) = load_data()

print('shape of x_train: ' + str(x_train.shape))
print('shape of y_train: ' + str(y_train.shape))
print('shape of x_test: ' + str(x_test.shape))
print('shape of y_test: ' + str(y_test.shape))
print('number of classes: ' + str(np.max(y_train) - np.min(y_train) + 1))
```

```
shape of x_train: (50000, 32, 32, 3)
shape of y_train: (50000, 1)
shape of x_test: (10000, 32, 32, 3)
shape of y_test: (10000, 1)
number of classes: 10
```

In [3]:

```
n_train = x_train.shape[0]
n_test = x_test.shape[0]
n_cls = np.max(y_train) - np.min(y_train) + 1

rand_perm = np.random.permutation(n_train)

x_train = x_train[rand_perm]
y_train = y_train[rand_perm]

# x_train = x_train[..., np.newaxis]

y_train = tf.one_hot(y_train, depth=n_cls, on_value=1, off_value=0)
y_test = tf.one_hot(y_test, depth=n_cls, on_value=1, off_value=0)

y_train = np.moveaxis(y_train, 2, 1)
y_test = np.moveaxis(y_test, 2, 1)

y_train = np.squeeze(y_train, axis=2)
y_test = np.squeeze(y_test, axis=2)

X = x_train[:40000]
y = y_train[:40000]

x_val = x_train[40000:]
y_val = y_train[40000:]

train = tf.data.Dataset.from_tensor_slices((x_train, y_train))
validate = tf.data.Dataset.from_tensor_slices((x_val, y_val))
test = tf.data.Dataset.from_tensor_slices((x_test, y_test))
```

In [4]:

```
print('Shape of x_train: ' + str(X.shape))
print('Shape of y_train: ' + str(y.shape))
print('Shape of x_val: ' + str(x_val.shape))
print('Shape of y_val: ' + str(y_val.shape))
```

```
Shape of x_train: (40000, 32, 32, 3)
Shape of y_train: (40000, 10)
Shape of x_val: (10000, 32, 32, 3)
Shape of y_val: (10000, 10)
```

In [5]:

```
IMG_SIZE=32

resize_and_rescale = tf.keras.Sequential([
    layers.experimental.preprocessing.Resizing(IMG_SIZE, IMG_SIZE),
    layers.experimental.preprocessing.Rescaling(1./255)
])
```

In [6]:

```
data_augmentation = tf.keras.Sequential([
    layers.experimental.preprocessing.RandomFlip("horizontal_and_vertical"),
    layers.experimental.preprocessing.RandomRotation(0.2),
    layers.experimental.preprocessing.RandomContrast(0.7)
])
```

In [7]:

```
batch_size = 32
AUTOTUNE = tf.data.AUTOTUNE

def prepare(ds, shuffle=False, augment=False):
    # Resize and rescale all datasets
    ds = ds.map(lambda x, y: (resize_and_rescale(x), y),
                num_parallel_calls=AUTOTUNE)

    if shuffle:
        ds = ds.shuffle(1000)

    # Batch all datasets
    ds = ds.batch(batch_size)

    # Use data augmentation only on the training set
    if augment:
        ds = ds.map(lambda x, y: (data_augmentation(x, training=True), y),
                    num_parallel_calls=AUTOTUNE)

    # Use buffered prefecting on all datasets
    return ds.prefetch(buffer_size=AUTOTUNE)
```

In [8]:

```
train = prepare(train, shuffle=False, augment=False)
validate = prepare(validate)
test = prepare(test)

print(f"Train: {train}\n{len(train)*batch_size}")
print(f"Val: {validate}\n{len(validate)*batch_size}")
print(f"Test: {test}\n{len(test)*batch_size}")
```

```
Train: <PrefetchDataset shapes: ((None, 32, 32, 3), (None, 10)), types: (tf.float32, tf.int32)>
50016
Val: <PrefetchDataset shapes: ((None, 32, 32, 3), (None, 10)), types: (tf.float32, tf.int32)>
10016
Test: <PrefetchDataset shapes: ((None, 32, 32, 3), (None, 10)), types: (tf.float32, tf.int32)>
10016
```

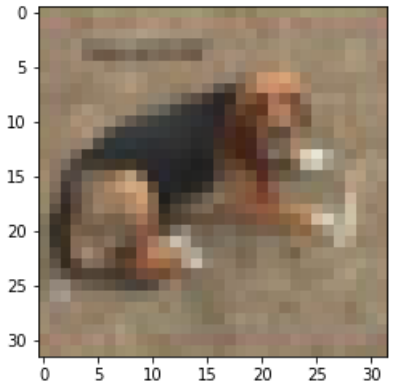
In [9]:

```
image, label = next(iter(train))
print(label[0])
pt.imshow(image[0,...], cmap='gray')
```

```
tf.Tensor([0 0 0 0 0 1 0 0 0 0], shape=(10,), dtype=int32)
```

Out[9]:

```
<matplotlib.image.AxesImage at 0x7f8aeb3f0c10>
```




In [10]:

```
inp = layers.Input(shape=(32, 32, 3))

x = layers.Conv2D(16, (5, 5), padding="same")(inp)
```

```
y = layers.BatchNormalization()(x)
y = layers.Activation(layers.LeakyReLU())(y)

#-------------------------------------------------------
x = layers.Conv2D(16, (5, 5), padding="same")(y)
y = layers.BatchNormalization()(x)
y = layers.Activation(layers.LeakyReLU())(y)
y = layers.Conv2D(16, (5, 5), padding='same')(y)
y = layers.BatchNormalization()(y)
y = layers.Activation(layers.LeakyReLU())(y)

x = layers.Add()([x, y])
#-------------------------------------------------------
x = layers.Conv2D(16, (5, 5), padding="same")(x)
y = layers.BatchNormalization()(x)
y = layers.Activation(layers.LeakyReLU())(y)
y = layers.Conv2D(16, (5, 5), padding='same')(y)
y = layers.BatchNormalization()(y)
y = layers.Activation(layers.LeakyReLU())(y)

x = layers.Add()([x, y])

x = layers.MaxPooling2D((2, 2))(x)
#-------------------------------------------------------
x = layers.Conv2D(32, (5, 5), padding="same")(x)
y = layers.BatchNormalization()(x)
y = layers.Activation(layers.LeakyReLU())(y)
y = layers.Conv2D(32, (5, 5), padding='same')(y)
y = layers.BatchNormalization()(y)
y = layers.Activation(layers.LeakyReLU())(y)

x = layers.Add()([x, y])
#-------------------------------------------------------
x = layers.Conv2D(32, (3, 3), padding="same")(x)
y = layers.BatchNormalization()(x)
y = layers.Activation(layers.LeakyReLU())(y)
y = layers.Conv2D(32, (3, 3), padding='same')(y)
y = layers.BatchNormalization()(y)
y = layers.Activation(layers.LeakyReLU())(y)

x = layers.Add()([x, y])
#-------------------------------------------------------
x = layers.Conv2D(32, (3, 3), padding="same")(x)
y = layers.BatchNormalization()(x)
y = layers.Activation(layers.LeakyReLU())(y)
y = layers.Conv2D(32, (3, 3), padding='same')(y)
y = layers.BatchNormalization()(y)
y = layers.Activation(layers.LeakyReLU())(y)

x = layers.Add()([x, y])

x = layers.MaxPooling2D((2,2))(x)
#-------------------------------------------------------
x = layers.Conv2D(64, (3, 3), padding="same")(x)
y = layers.BatchNormalization()(x)
y = layers.Activation(layers.LeakyReLU())(y)
y = layers.Conv2D(64, (3, 3), padding='same')(y)
y = layers.BatchNormalization()(y)
y = layers.Activation(layers.LeakyReLU())(y)

x = layers.Add()([x, y])
#-------------------------------------------------------
x = layers.Conv2D(64, (3, 3), padding="same")(x)
y = layers.BatchNormalization()(x)
y = layers.Activation(layers.LeakyReLU())(y)
y = layers.Conv2D(64, (3, 3), padding='same')(y)
y = layers.BatchNormalization()(y)
y = layers.Activation(act.relu)(y)

x = layers.Add()([x, y])
#-------------------------------------------------------
x = layers.Conv2D(64, (3, 3), padding="same")(x)
y = layers.BatchNormalization()(x)
y = layers.Activation(layers.LeakyReLU())(y)
y = layers.Conv2D(64, (3, 3), padding='same')(y)
y = layers.BatchNormalization()(y)
```

```
y = layers.Activation(act.relu)(y)

x = layers.Add()([x, y])
#----------------------------------------------------------

x = layers.AveragePooling2D((2, 2))(x)

z = layers.Flatten()(x)
z = layers.Dense(128)(z)
z = layers.Activation(layers.LeakyReLU())(z)

z = layers.Dropout(0.25)(z)

z = layers.Dense(10)(z)
z = layers.Activation(act.softmax)(z)
```

In [11]:

```
model = tf.keras.models.Model(inputs=inp, outputs=z)
```

In [12]:

```
model.summary()
```

```
Model: "model"
```

| Layer (type) | Output Shape | Param # | Connected to |
|---|---|---|---|
| input_1 (InputLayer) | [(None, 32, 32, 3)] | 0 | |
| conv2d (Conv2D) | (None, 32, 32, 16) | 1216 | input_1[0][0] |
| batch_normalization (BatchNorma | (None, 32, 32, 16) | 64 | conv2d[0][0] |
| activation (Activation) | (None, 32, 32, 16) | 0 | batch_normalization[0][0] |
| conv2d_1 (Conv2D) | (None, 32, 32, 16) | 6416 | activation[0][0] |
| batch_normalization_1 (BatchNor | (None, 32, 32, 16) | 64 | conv2d_1[0][0] |
| activation_1 (Activation) | (None, 32, 32, 16) | 0 | batch_normalization_1[0][0] |
| conv2d_2 (Conv2D) | (None, 32, 32, 16) | 6416 | activation_1[0][0] |
| batch_normalization_2 (BatchNor | (None, 32, 32, 16) | 64 | conv2d_2[0][0] |
| activation_2 (Activation) | (None, 32, 32, 16) | 0 | batch_normalization_2[0][0] |
| add (Add) | (None, 32, 32, 16) | 0 | conv2d_1[0][0]<br>activation_2[0][0] |
| conv2d_3 (Conv2D) | (None, 32, 32, 16) | 6416 | add[0][0] |
| batch_normalization_3 (BatchNor | (None, 32, 32, 16) | 64 | conv2d_3[0][0] |
| activation_3 (Activation) | (None, 32, 32, 16) | 0 | batch_normalization_3[0][0] |
| conv2d_4 (Conv2D) | (None, 32, 32, 16) | 6416 | activation_3[0][0] |
| batch_normalization_4 (BatchNor | (None, 32, 32, 16) | 64 | conv2d_4[0][0] |
| activation_4 (Activation) | (None, 32, 32, 16) | 0 | batch_normalization_4[0][0] |
| add_1 (Add) | (None, 32, 32, 16) | 0 | conv2d_3[0][0]<br>activation_4[0][0] |
| max_pooling2d (MaxPooling2D) | (None, 16, 16, 16) | 0 | add_1[0][0] |
| conv2d_5 (Conv2D) | (None, 16, 16, 32) | 12832 | max_pooling2d[0][0] |
| batch_normalization_5 (BatchNor | (None, 16, 16, 32) | 128 | conv2d_5[0][0] |
| activation_5 (Activation) | (None, 16, 16, 32) | 0 | batch_normalization_5[0][0] |
| conv2d_6 (Conv2D) | (None, 16, 16, 32) | 25632 | activation_5[0][0] |
| batch_normalization_6 (BatchNor | (None, 16, 16, 32) | 128 | conv2d_6[0][0] |

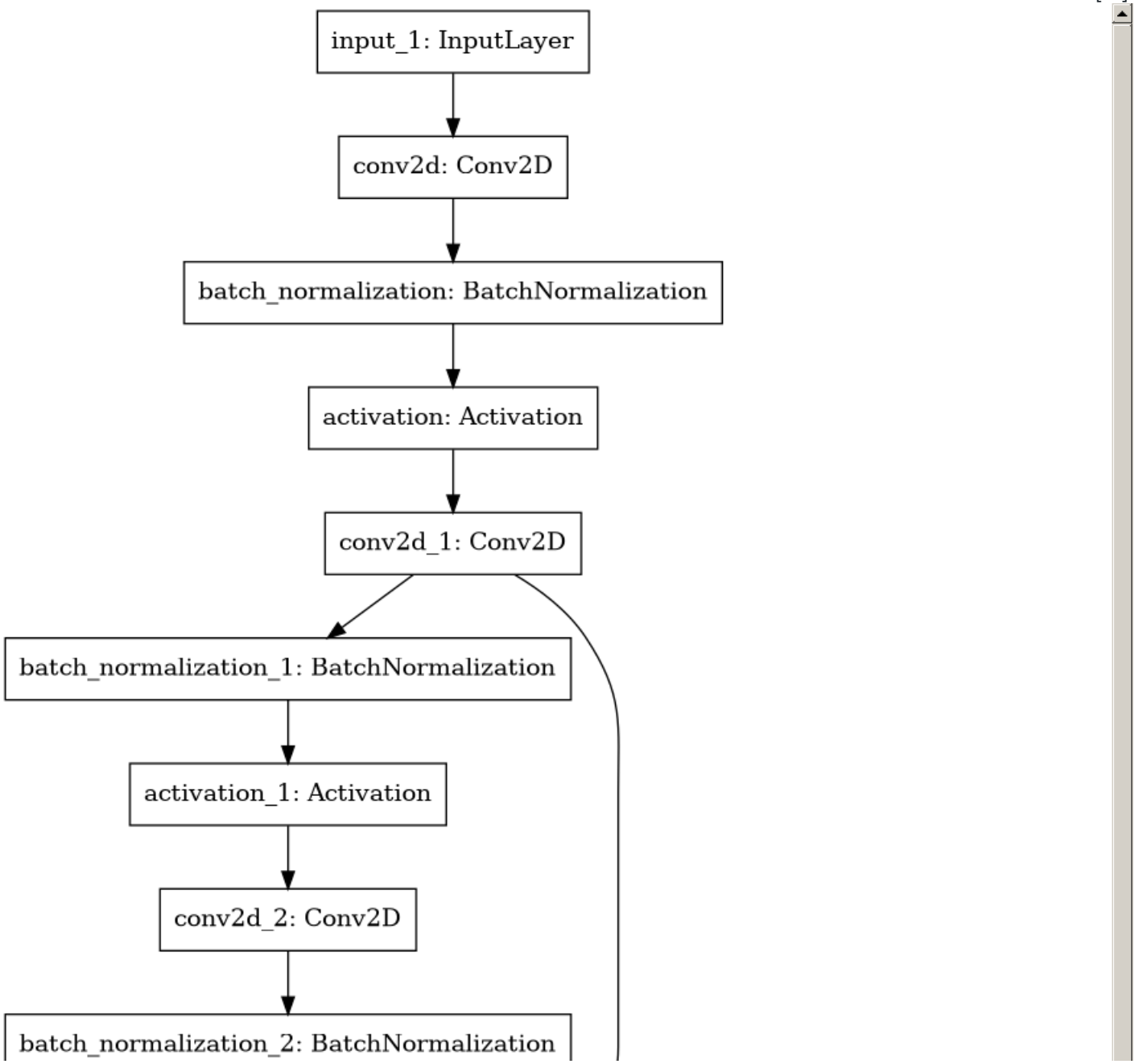
| | | | |
|---|---|---|---|
| activation_6 (Activation) | (None, 16, 16, 32) | 0 | batch_normalization_6[0][0] |
| add_2 (Add) | (None, 16, 16, 32) | 0 | conv2d_5[0][0]<br>activation_6[0][0] |
| conv2d_7 (Conv2D) | (None, 16, 16, 32) | 9248 | add_2[0][0] |
| batch_normalization_7 (BatchNor | (None, 16, 16, 32) | 128 | conv2d_7[0][0] |
| activation_7 (Activation) | (None, 16, 16, 32) | 0 | batch_normalization_7[0][0] |
| conv2d_8 (Conv2D) | (None, 16, 16, 32) | 9248 | activation_7[0][0] |
| batch_normalization_8 (BatchNor | (None, 16, 16, 32) | 128 | conv2d_8[0][0] |
| activation_8 (Activation) | (None, 16, 16, 32) | 0 | batch_normalization_8[0][0] |
| add_3 (Add) | (None, 16, 16, 32) | 0 | conv2d_7[0][0]<br>activation_8[0][0] |
| conv2d_9 (Conv2D) | (None, 16, 16, 32) | 9248 | add_3[0][0] |
| batch_normalization_9 (BatchNor | (None, 16, 16, 32) | 128 | conv2d_9[0][0] |
| activation_9 (Activation) | (None, 16, 16, 32) | 0 | batch_normalization_9[0][0] |
| conv2d_10 (Conv2D) | (None, 16, 16, 32) | 9248 | activation_9[0][0] |
| batch_normalization_10 (BatchNo | (None, 16, 16, 32) | 128 | conv2d_10[0][0] |
| activation_10 (Activation) | (None, 16, 16, 32) | 0 | batch_normalization_10[0][0] |
| add_4 (Add) | (None, 16, 16, 32) | 0 | conv2d_9[0][0]<br>activation_10[0][0] |
| max_pooling2d_1 (MaxPooling2D) | (None, 8, 8, 32) | 0 | add_4[0][0] |
| conv2d_11 (Conv2D) | (None, 8, 8, 64) | 18496 | max_pooling2d_1[0][0] |
| batch_normalization_11 (BatchNo | (None, 8, 8, 64) | 256 | conv2d_11[0][0] |
| activation_11 (Activation) | (None, 8, 8, 64) | 0 | batch_normalization_11[0][0] |
| conv2d_12 (Conv2D) | (None, 8, 8, 64) | 36928 | activation_11[0][0] |
| batch_normalization_12 (BatchNo | (None, 8, 8, 64) | 256 | conv2d_12[0][0] |
| activation_12 (Activation) | (None, 8, 8, 64) | 0 | batch_normalization_12[0][0] |
| add_5 (Add) | (None, 8, 8, 64) | 0 | conv2d_11[0][0]<br>activation_12[0][0] |
| conv2d_13 (Conv2D) | (None, 8, 8, 64) | 36928 | add_5[0][0] |
| batch_normalization_13 (BatchNo | (None, 8, 8, 64) | 256 | conv2d_13[0][0] |
| activation_13 (Activation) | (None, 8, 8, 64) | 0 | batch_normalization_13[0][0] |
| conv2d_14 (Conv2D) | (None, 8, 8, 64) | 36928 | activation_13[0][0] |
| batch_normalization_14 (BatchNo | (None, 8, 8, 64) | 256 | conv2d_14[0][0] |
| activation_14 (Activation) | (None, 8, 8, 64) | 0 | batch_normalization_14[0][0] |
| add_6 (Add) | (None, 8, 8, 64) | 0 | conv2d_13[0][0]<br>activation_14[0][0] |
| conv2d_15 (Conv2D) | (None, 8, 8, 64) | 36928 | add_6[0][0] |
| batch_normalization_15 (BatchNo | (None, 8, 8, 64) | 256 | conv2d_15[0][0] |
| activation_15 (Activation) | (None, 8, 8, 64) | 0 | batch_normalization_15[0][0] |
| conv2d_16 (Conv2D) | (None, 8, 8, 64) | 36928 | activation_15[0][0] |
| batch_normalization_16 (BatchNo | (None, 8, 8, 64) | 256 | conv2d_16[0][0] |

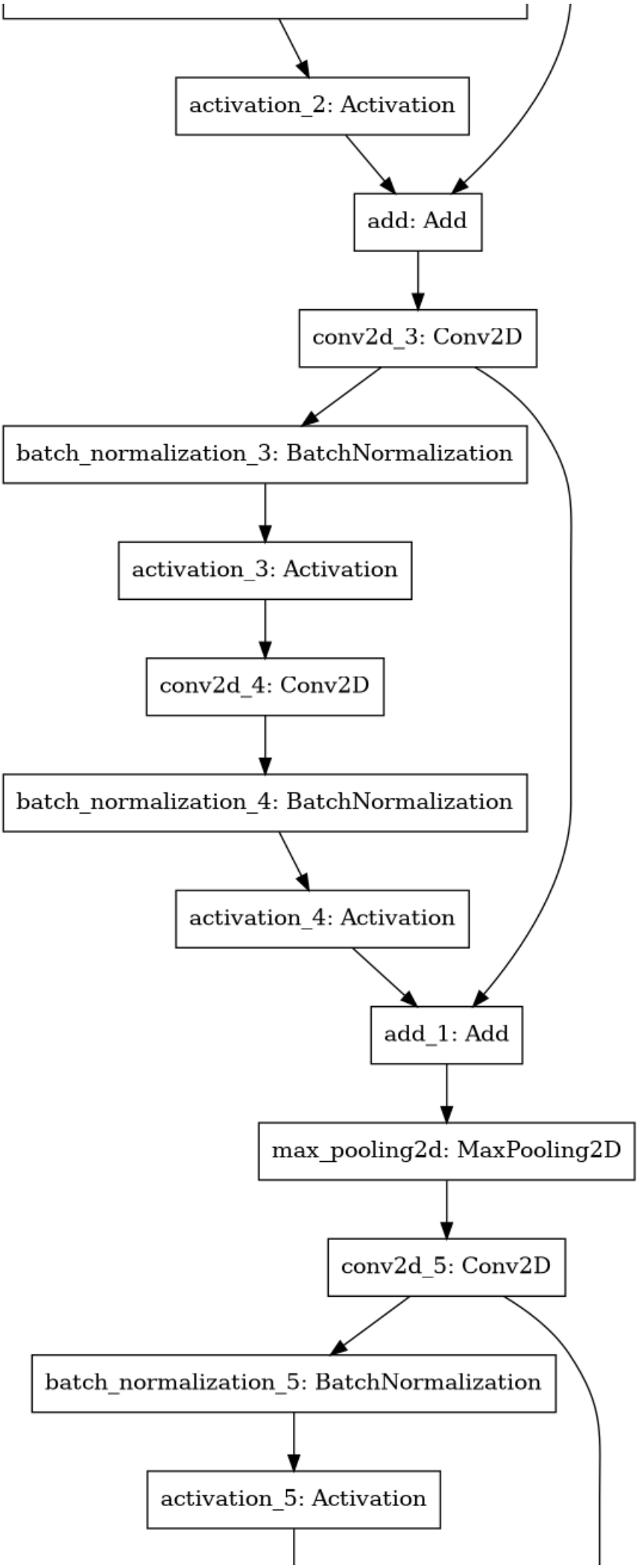
```
activation_16 (Activation)      (None, 8, 8, 64)     0           batch_normalization_16[0][0]
__________________________________________________________________________________________________
add_7 (Add)                     (None, 8, 8, 64)     0           conv2d_15[0][0]
                                                                 activation_16[0][0]
__________________________________________________________________________________________________
average_pooling2d (AveragePooli (None, 4, 4, 64)     0           add_7[0][0]
__________________________________________________________________________________________________
flatten (Flatten)               (None, 1024)         0           average_pooling2d[0][0]
__________________________________________________________________________________________________
dense (Dense)                   (None, 128)          131200      flatten[0][0]
__________________________________________________________________________________________________
activation_17 (Activation)      (None, 128)          0           dense[0][0]
__________________________________________________________________________________________________
dropout (Dropout)               (None, 128)          0           activation_17[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 10)           1290        dropout[0][0]
__________________________________________________________________________________________________
activation_18 (Activation)      (None, 10)           0           dense_1[0][0]
==================================================================================================
Total params: 440,586
Trainable params: 439,274
Non-trainable params: 1,312
__________________________________________________________________________________________________
```

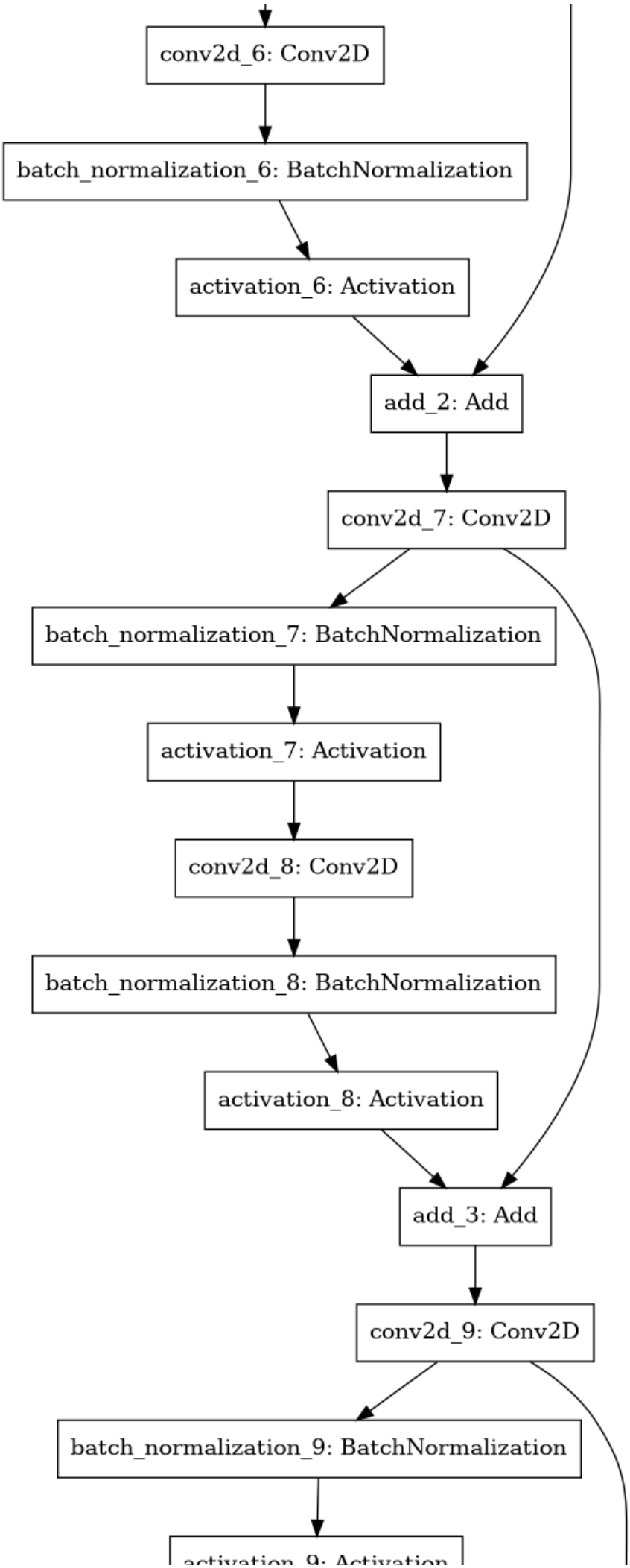
In [13]:

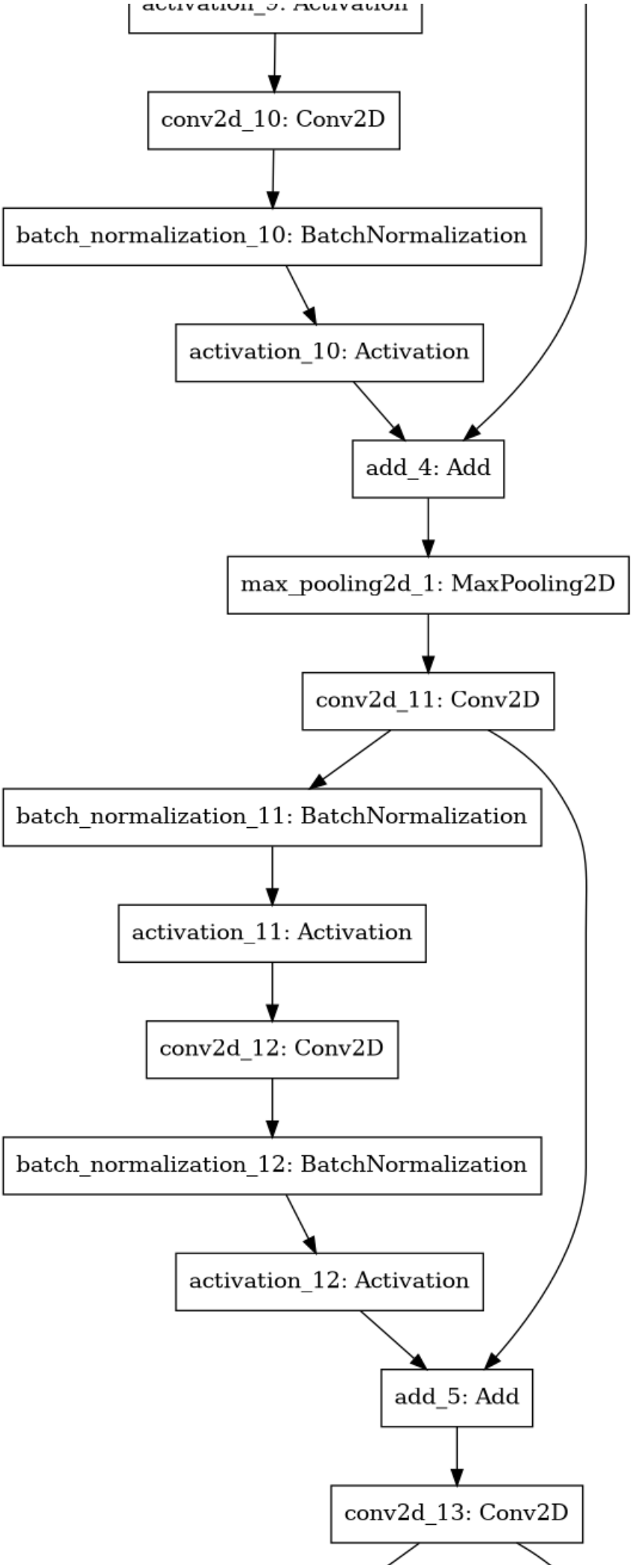
```
tf.keras.utils.plot_model(model, to_file="skip_connection_graph.png")
```

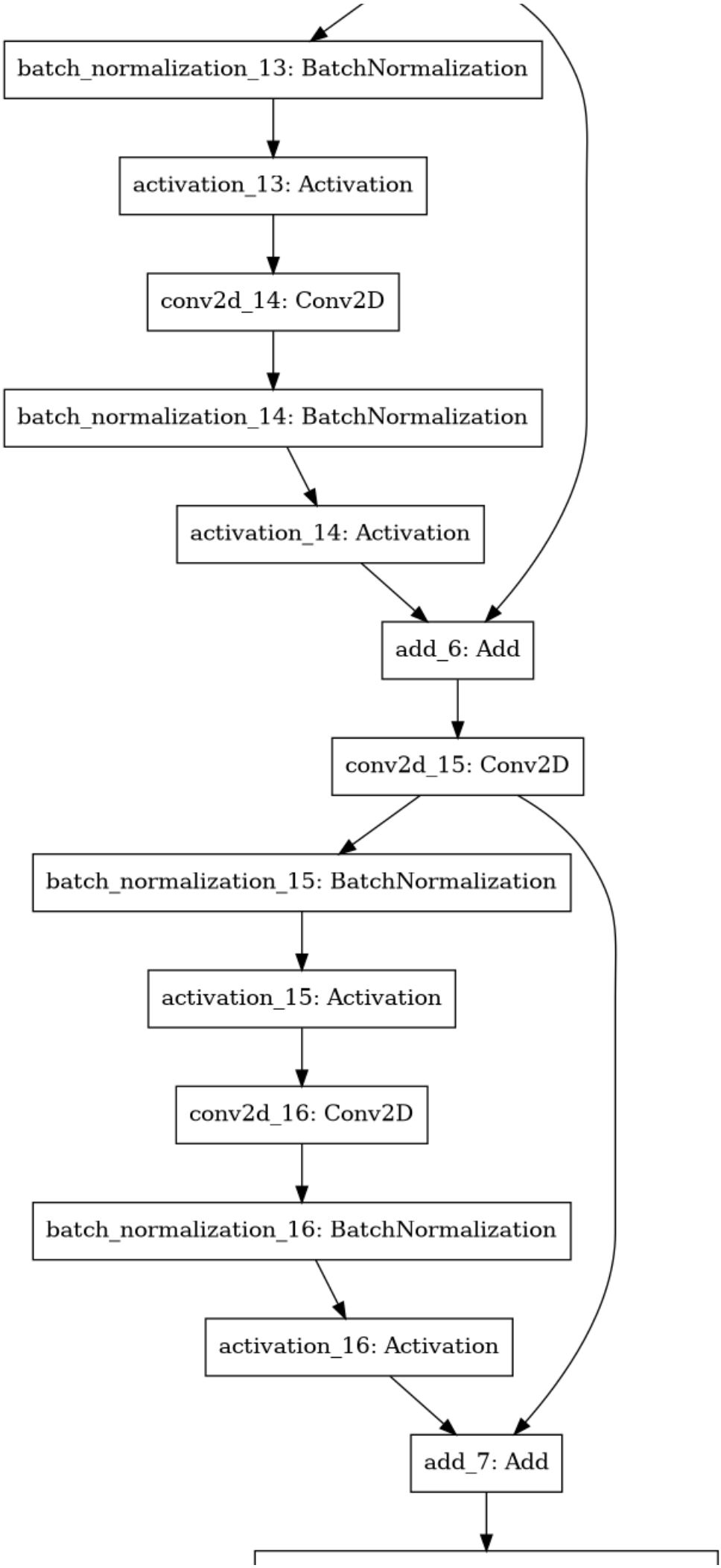
Out[13]:

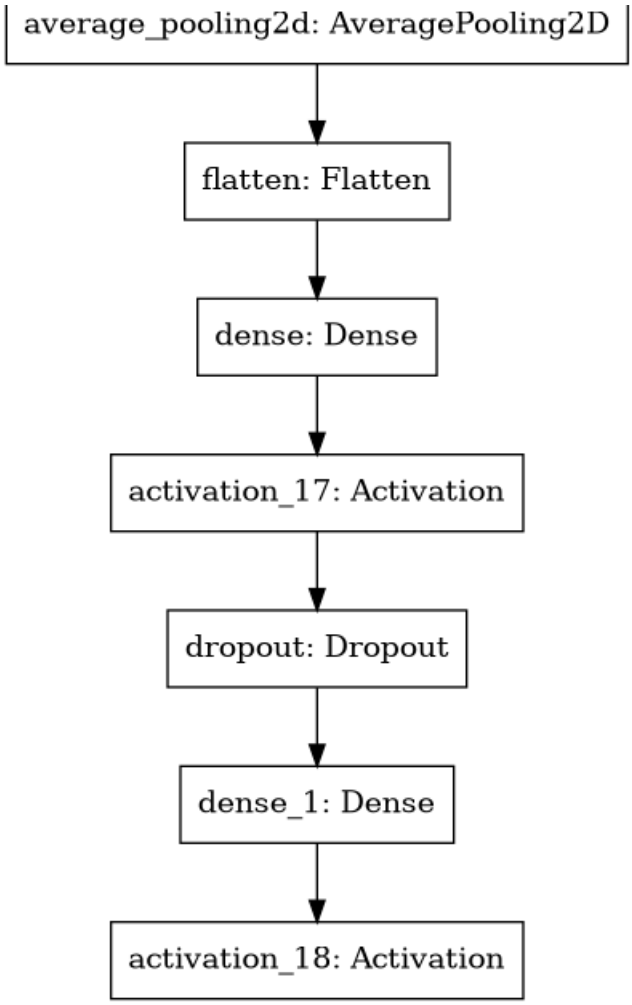



activation_2: Activation
add: Add
conv2d_3: Conv2D
batch_normalization_3: BatchNormalization
activation_3: Activation
conv2d_4: Conv2D
batch_normalization_4: BatchNormalization
activation_4: Activation
add_1: Add
max_pooling2d: MaxPooling2D
conv2d_5: Conv2D
batch_normalization_5: BatchNormalization
activation_5: Activation


conv2d_6: Conv2D
batch_normalization_6: BatchNormalization
activation_6: Activation
add_2: Add
conv2d_7: Conv2D
batch_normalization_7: BatchNormalization
activation_7: Activation
conv2d_8: Conv2D
batch_normalization_8: BatchNormalization
activation_8: Activation
add_3: Add
conv2d_9: Conv2D
batch_normalization_9: BatchNormalization


conv2d_10: Conv2D
batch_normalization_10: BatchNormalization
activation_10: Activation
add_4: Add
max_pooling2d_1: MaxPooling2D
conv2d_11: Conv2D
batch_normalization_11: BatchNormalization
activation_11: Activation
conv2d_12: Conv2D
batch_normalization_12: BatchNormalization
activation_12: Activation
add_5: Add
conv2d_13: Conv2D


batch_normalization_13: BatchNormalization
activation_13: Activation
conv2d_14: Conv2D
batch_normalization_14: BatchNormalization
activation_14: Activation
add_6: Add
conv2d_15: Conv2D
batch_normalization_15: BatchNormalization
activation_15: Activation
conv2d_16: Conv2D
batch_normalization_16: BatchNormalization
activation_16: Activation
add_7: Add




In [21]:

```
model.compile(tf.keras.optimizers.RMSprop(lr=0.0008),
              loss='categorical_crossentropy',
               metrics=['accuracy'])
# we have an extremely balanced dataset so classification accuracy is valid
```

In [23]:

```
history = model.fit(train,
          epochs=5,
          validation_data=validate)
```

```
Epoch 1/5
1563/1563 [==============================] - 24s 14ms/step - loss: 1.9479 - accuracy: 0.3814 - val_loss:
1.4315 - val_accuracy: 0.5058
Epoch 2/5
1563/1563 [==============================] - 21s 13ms/step - loss: 1.1864 - accuracy: 0.5866 - val_loss:
0.9536 - val_accuracy: 0.6662
Epoch 3/5
1563/1563 [==============================] - 21s 13ms/step - loss: 0.9929 - accuracy: 0.6529 - val_loss:
0.8586 - val_accuracy: 0.6940
Epoch 4/5
1563/1563 [==============================] - 21s 14ms/step - loss: 0.8849 - accuracy: 0.6888 - val_loss:
0.7403 - val_accuracy: 0.7444
Epoch 5/5
1563/1563 [==============================] - 21s 14ms/step - loss: 0.8109 - accuracy: 0.7132 - val_loss:
0.6696 - val_accuracy: 0.7639
```

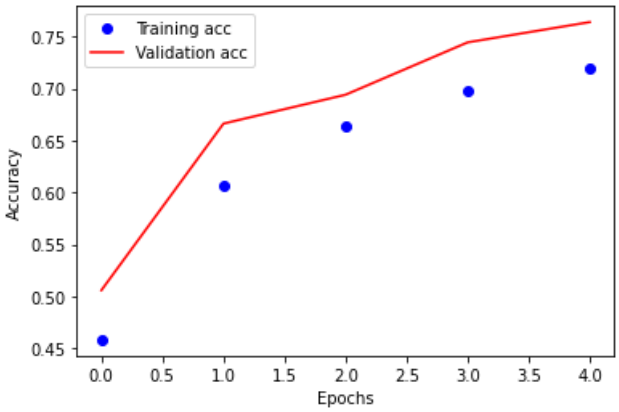
In [25]:

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

epochs = range(len(acc))

pt.plot(epochs, acc, 'bo', label='Training acc')
pt.plot(epochs, val_acc, 'r', label='Validation acc')
pt.xlabel('Epochs')
pt.ylabel('Accuracy')
pt.legend()
```

```
pt.show()
```




In [26]:

```
full_train = tf.data.Dataset.from_tensor_slices((x_train, y_train))
print(len(full_train))
full_batched_train = full_train.batch(32)
```

```
50000
```

In [27]:

```
model.compile(tf.keras.optimizers.RMSprop(lr=0.0001),
              loss='categorical_crossentropy',
               metrics=['accuracy'])
# we have an extremely balanced dataset so classification accuracy is valid
```

In [28]:

```
history = model.fit(full_batched_train,
          epochs=15)
```

```
Epoch 1/15
1563/1563 [==============================] - 20s 12ms/step - loss: 0.6242 - accuracy: 0.7854
Epoch 2/15
1563/1563 [==============================] - 18s 12ms/step - loss: 0.4885 - accuracy: 0.8285
Epoch 3/15
1563/1563 [==============================] - 18s 12ms/step - loss: 0.4197 - accuracy: 0.8549
Epoch 4/15
1563/1563 [==============================] - 18s 12ms/step - loss: 0.3614 - accuracy: 0.8763
Epoch 5/15
1563/1563 [==============================] - 19s 12ms/step - loss: 0.3082 - accuracy: 0.8949
Epoch 6/15
1563/1563 [==============================] - ETA: 0s - loss: 0.2596 - accuracy: 0.91 - 19s 12ms/step - l
oss: 0.2595 - accuracy: 0.9123
Epoch 7/15
1563/1563 [==============================] - 19s 12ms/step - loss: 0.2111 - accuracy: 0.9292
Epoch 8/15
1563/1563 [==============================] - 19s 12ms/step - loss: 0.1718 - accuracy: 0.9436
Epoch 9/15
1563/1563 [==============================] - 19s 12ms/step - loss: 0.1383 - accuracy: 0.9558
Epoch 10/15
1563/1563 [==============================] - 20s 13ms/step - loss: 0.1137 - accuracy: 0.9645
Epoch 11/15
1563/1563 [==============================] - 19s 12ms/step - loss: 0.0876 - accuracy: 0.9732
Epoch 12/15
1563/1563 [==============================] - 20s 13ms/step - loss: 0.0723 - accuracy: 0.9774
Epoch 13/15
1563/1563 [==============================] - 20s 13ms/step - loss: 0.0635 - accuracy: 0.9800
Epoch 14/15
1563/1563 [==============================] - 20s 13ms/step - loss: 0.0549 - accuracy: 0.9817
Epoch 15/15
1563/1563 [==============================] - 20s 13ms/step - loss: 0.0471 - accuracy: 0.9840
```

In [29]:

```
results = model.evaluate(x_test, y_test, batch_size=32)
print(f"Loss = {results[0]}")
print(f"Accuracy = {results[1]}")
```

```
313/313 [==============================] - 2s 4ms/step - loss: 1.4954 - accuracy: 0.7715
Loss = 1.4953542947769165
Accuracy = 0.7714999914169312
```