Introduction

In this notebook, I've used **CNN** to perform Image Classification on the Brain Tumor dataset. Since this dataset is small, if we train a neural network to it, it won't really give us a good result. Therefore, I'm going to use the concept of **Transfer Learning** and also custom CNN to train the model to compare accurate results.

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

```
Mounted at /content/drive
```

Note: Before you run, Import the dataset into your google drive and then try updating the paths specific to yout drive.

Dataset:

https://drive.google.com/drive/folders/1olI_u9_wkqnuJeqqmMxwxaF0041Eswtw?usp=sharing

Importing Libraries

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import cv2
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tqdm import tqdm
import os
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split
from tensorflow.keras.applications import EfficientNetB0
from tensorflow.keras.callbacks import EarlyStopping,
ReduceLROnPlateau, TensorBoard, ModelCheckpoint
from sklearn.metrics import classification_report,confusion_matrix
import ipywidgets as widgets
import io
from PIL import Image
from IPython.display import display,clear_output
from warnings import filterwarnings
# for dirname, _, filenames in
os.walk('/content/drive/MyDrive/Projects-for-Sale/Proj - 2 Skin
```

```
Disease Identification Using Image Analysis/dataset (skin diseases)'):
#     for filename in filenames:
#         print(os.path.join(dirname, filename))
```

Data Preperation

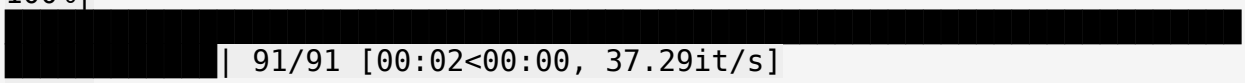
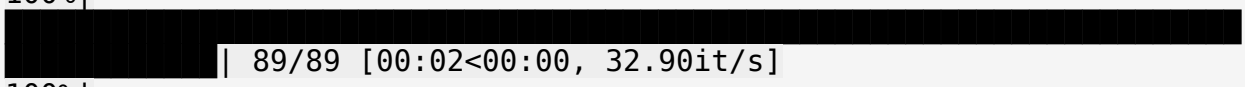
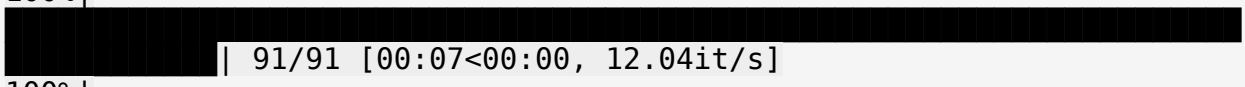
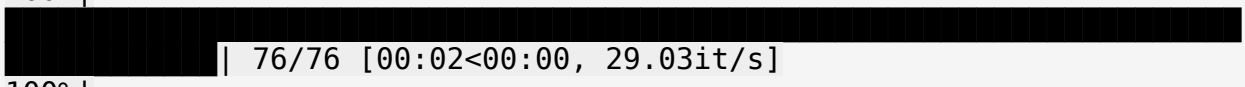
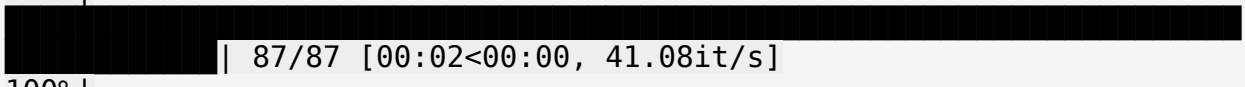
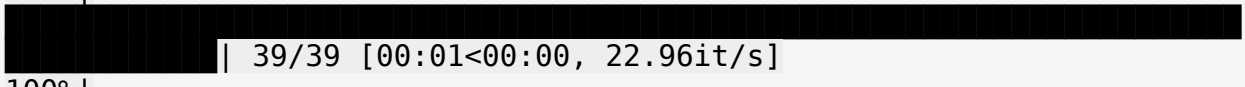
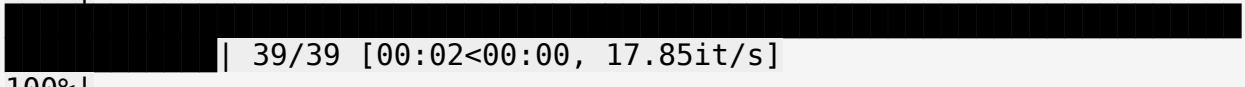
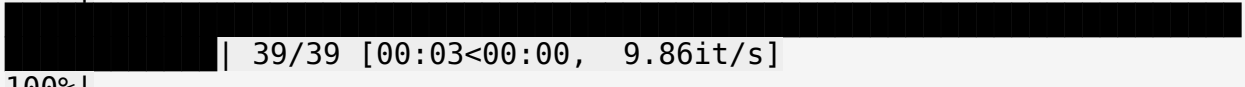
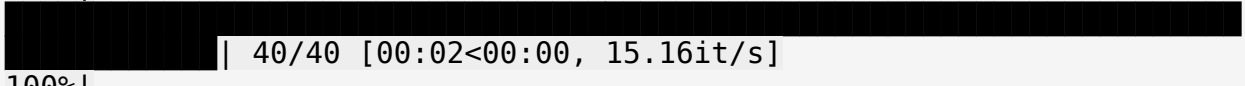
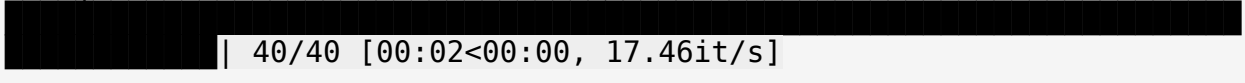
```
labels = ["actinic keratosis",
"dermatofibroma",
"melanoma",
"seborrheic keratosis",
"squamous cell carcinoma",
# "Acne_and_rosacea",
# "Eczema",
# "Tinea_Ringworm"
]
```

%pwd

```
'C:\\Users\\vsneh\\OneDrive\\Desktop\\Sale Projects\\Proj - 2 Skin Disease Identification Using Image Analysis\\Main Code'
```

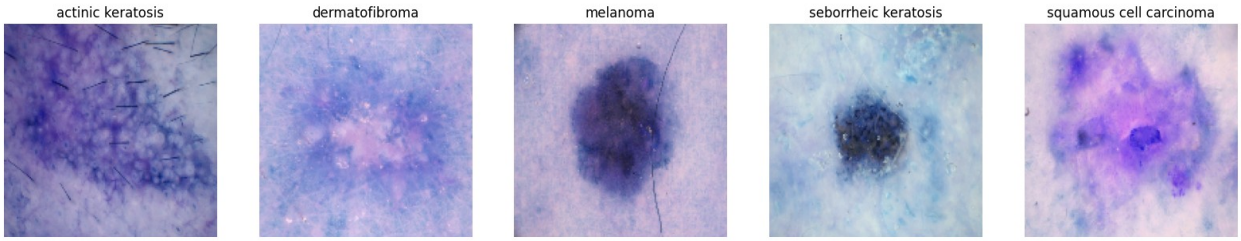
We start off by appending all the images from the directories into a Python list and then converting them into numpy arrays after resizing it.

```
X_train = []
y_train = []
image_size = 150
dir = '../dataset (skin diseases)'
for i in labels:
    folderPath = os.path.join(dir,'Train',i)
    for j in tqdm(os.listdir(folderPath)):
        img = cv2.imread(os.path.join(folderPath,j))
        img = cv2.resize(img,(image_size, image_size))
        X_train.append(img)
        y_train.append(i)

for i in labels:
    folderPath = os.path.join(dir,'Test',i)
    for j in tqdm(os.listdir(folderPath)):
        img = cv2.imread(os.path.join(folderPath,j))
        img = cv2.resize(img,(image_size,image_size))
        X_train.append(img)
        y_train.append(i)

X_train = np.array(X_train)
y_train = np.array(y_train)
```

```
100%|
```

```
| 91/91 [00:02<00:00, 37.29it/s]
100%|
```

```
| 89/89 [00:02<00:00, 32.90it/s]
100%|
```

```
| 91/91 [00:07<00:00, 12.04it/s]
100%|
```

```
| 76/76 [00:02<00:00, 29.03it/s]
100%|
```

```
| 87/87 [00:02<00:00, 41.08it/s]
100%|
```

```
| 39/39 [00:01<00:00, 22.96it/s]
100%|
```

```
| 39/39 [00:02<00:00, 17.85it/s]
100%|
```

```
| 39/39 [00:03<00:00,  9.86it/s]
100%|
```

```
| 40/40 [00:02<00:00, 15.16it/s]
100%|
```

```
| 40/40 [00:02<00:00, 17.46it/s]
```

```
k=0
fig, ax = plt.subplots(1,5,figsize=(20,20))
fig.text(s='Sample Image From Each Class ',size=18,fontweight='bold',
             fontname='monospace',y=0.62,x=0.4,alpha=0.8)
for i in labels:
    j=0
    while True :
        if y_train[j]==i:
            ax[k].imshow(X_train[j])
            ax[k].set_title(y_train[j])
            ax[k].axis('off')
            k+=1
            break
        j+=1
```

Sample Image From Each Class




```
X_train, y_train = shuffle(X_train,y_train, random_state=101)

X_train.shape
```

```
(631, 150, 150, 3)
```

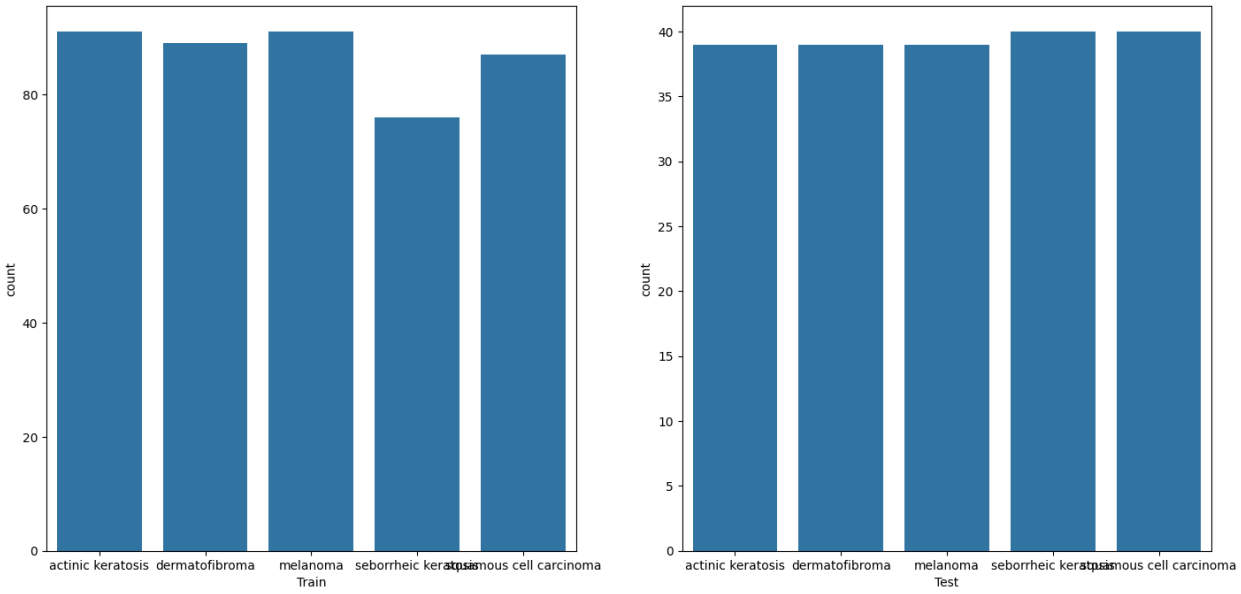
Dividing the dataset into **Training** and **Testing** sets.

```
X_train,X_test,y_train,y_test = train_test_split(X_train,y_train,
test_size=0.1,random_state=101)
```

Performing **One Hot Encoding** on the labels after converting it into numerical values:

```
y_train_new = []
for i in y_train:
    y_train_new.append(labels.index(i))
y_train = y_train_new
y_train = tf.keras.utils.to_categorical(y_train)


y_test_new = []
for i in y_test:
    y_test_new.append(labels.index(i))
y_test = y_test_new
y_test = tf.keras.utils.to_categorical(y_test)

train_labels = []
test_labels = []

img_size= 300

for i in os.listdir(dir+'/Train/'):
    for j in os.listdir(dir+"/Train/"+i):
       train_labels.append(i)

for i in os.listdir(dir+'/Test/'):
    for j in os.listdir(dir+"/Test/"+i):
        test_labels.append(i)
```

```
plt.figure(figsize = (17,8));
lis = ['Train', 'Test']
for i,j in enumerate([train_labels, test_labels]):
    plt.subplot(1,2, i+1);
    sns.countplot(x = j);
    plt.xlabel(lis[i])
```



Custom CNN

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Activation, Dropout, Flatten,
Dense, Conv2D, MaxPool2D, BatchNormalization
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator

import numpy as np
from glob import glob
import matplotlib.pyplot as plt

# Start training freshly
tf.keras.backend.clear_session()

model = Sequential()
model.add(Conv2D(filters=32,kernel_size=(3,3),input_shape=(image_size,
image_size,3),activation='relu'))
model.add(BatchNormalization())
model.add(MaxPool2D(pool_size=(2,2)))
```

```
model.add(Conv2D(filters=64,kernel_size=(3,3),activation='relu'))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(0.5))

model.add(Conv2D(filters=128,kernel_size=(3,3),activation='relu'))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(0.5))

model.add(Flatten())

# model.add(Dense(64))
# model.add(Activation('relu'))
# model.add(Dropout(0.5))

model.add(Dense(5))
model.add(Activation('softmax'))

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

model.summary()
```

```
WARNING:tensorflow:From C:\Users\vsneh\AppData\Roaming\Python\
Python310\site-packages\keras\src\backend\common\global_state.py:73:
The name tf.reset_default_graph is deprecated. Please use
tf.compat.v1.reset_default_graph instead.


C:\Users\vsneh\AppData\Roaming\Python\Python310\site-packages\keras\
src\layers\convolutional\base_conv.py:99: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(

Model: "sequential"
```

```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━
┓
┃ Layer (type)                         ┃ Output Shape                ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━
┩
│ conv2d (Conv2D)                      │ (None, 148, 148, 32)        │
896 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────
┤
│ batch_normalization                  │ (None, 148, 148, 32)        │
128 │
│ (BatchNormalization)                 │                             │
```

```
│
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ max_pooling2d (MaxPooling2D)         │ (None, 74, 74, 32)          │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ conv2d_1 (Conv2D)                    │ (None, 72, 72, 64)          │
18,496 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ max_pooling2d_1 (MaxPooling2D)       │ (None, 36, 36, 64)          │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ dropout (Dropout)                    │ (None, 36, 36, 64)          │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ conv2d_2 (Conv2D)                    │ (None, 34, 34, 128)         │
73,856 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ max_pooling2d_2 (MaxPooling2D)       │ (None, 17, 17, 128)         │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ dropout_1 (Dropout)                  │ (None, 17, 17, 128)         │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ flatten (Flatten)                    │ (None, 36992)               │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ dense (Dense)                        │ (None, 5)                   │
184,965 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ activation (Activation)              │ (None, 5)                   │
0 │
└──────────────────────────────────────┴─────────────────────────────┴
─────────────────┘

 Total params: 278,341 (1.06 MB)

 Trainable params: 278,277 (1.06 MB)

 Non-trainable params: 64 (256.00 B)
```

```
tensorboard = TensorBoard(log_dir = 'logs')
checkpoint = ModelCheckpoint("skin-
snehit.keras",monitor="val_accuracy",save_best_only=True,mode="auto",v
erbose=1)
reduce_lr = ReduceLROnPlateau(monitor = 'val_accuracy', factor = 0.3,
patience = 2, min_delta = 0.001,
                              mode='auto',verbose=1)

history = model.fit(X_train,y_train,validation_split=0.1, epochs =60,
verbose=1, batch_size=16,
                    callbacks=[tensorboard,checkpoint,reduce_lr])

Epoch 1/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 372ms/step - accuracy: 0.2381 - loss:
6.4799
Epoch 1: val_accuracy improved from -inf to 0.28070, saving model to
skin-snehit.keras
32/32 ━━━━━━━━━━━━━━━━━━━━ 18s 421ms/step - accuracy: 0.2410 - loss:
6.3864 - val_accuracy: 0.2807 - val_loss: 1.5366 - learning_rate:
0.0010
Epoch 2/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 372ms/step - accuracy: 0.4570 - loss:
1.1745
Epoch 2: val_accuracy improved from 0.28070 to 0.40351, saving model
to skin-snehit.keras
32/32 ━━━━━━━━━━━━━━━━━━━━ 13s 394ms/step - accuracy: 0.4574 - loss:
1.1728 - val_accuracy: 0.4035 - val_loss: 1.4255 - learning_rate:
0.0010
Epoch 3/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 371ms/step - accuracy: 0.5698 - loss:
0.9692
Epoch 3: val_accuracy improved from 0.40351 to 0.45614, saving model
to skin-snehit.keras
32/32 ━━━━━━━━━━━━━━━━━━━━ 12s 384ms/step - accuracy: 0.5708 - loss:
0.9670 - val_accuracy: 0.4561 - val_loss: 1.2560 - learning_rate:
0.0010
Epoch 4/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 372ms/step - accuracy: 0.6645 - loss:
0.8333
Epoch 4: val_accuracy improved from 0.45614 to 0.54386, saving model
to skin-snehit.keras
32/32 ━━━━━━━━━━━━━━━━━━━━ 12s 388ms/step - accuracy: 0.6638 - loss:
0.8349 - val_accuracy: 0.5439 - val_loss: 1.2798 - learning_rate:
0.0010
Epoch 5/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 380ms/step - accuracy: 0.7418 - loss:
0.6856
Epoch 5: val_accuracy did not improve from 0.54386
32/32 ━━━━━━━━━━━━━━━━━━━━ 13s 391ms/step - accuracy: 0.7419 - loss:
0.6854 - val_accuracy: 0.4035 - val_loss: 1.8791 - learning_rate:
```

```
0.0010
Epoch 6/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 381ms/step - accuracy: 0.7702 - loss:
0.5527
Epoch 6: val_accuracy improved from 0.54386 to 0.56140, saving model
to skin-snehit.keras
32/32 ━━━━━━━━━━━━━━━━━━━━ 13s 400ms/step - accuracy: 0.7702 - loss:
0.5535 - val_accuracy: 0.5614 - val_loss: 1.0702 - learning_rate:
0.0010
Epoch 7/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 371ms/step - accuracy: 0.8178 - loss:
0.5252
Epoch 7: val_accuracy improved from 0.56140 to 0.57895, saving model
to skin-snehit.keras
32/32 ━━━━━━━━━━━━━━━━━━━━ 13s 389ms/step - accuracy: 0.8169 - loss:
0.5277 - val_accuracy: 0.5789 - val_loss: 1.0871 - learning_rate:
0.0010
Epoch 8/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 388ms/step - accuracy: 0.7708 - loss:
0.5849
Epoch 8: val_accuracy improved from 0.57895 to 0.59649, saving model
to skin-snehit.keras
32/32 ━━━━━━━━━━━━━━━━━━━━ 13s 406ms/step - accuracy: 0.7714 - loss:
0.5838 - val_accuracy: 0.5965 - val_loss: 1.2169 - learning_rate:
0.0010
Epoch 9/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 362ms/step - accuracy: 0.7976 - loss:
0.5045
Epoch 9: val_accuracy did not improve from 0.59649
32/32 ━━━━━━━━━━━━━━━━━━━━ 12s 375ms/step - accuracy: 0.7980 - loss:
0.5035 - val_accuracy: 0.5789 - val_loss: 1.1823 - learning_rate:
0.0010
Epoch 10/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 409ms/step - accuracy: 0.8766 - loss:
0.3231
Epoch 10: val_accuracy improved from 0.59649 to 0.63158, saving model
to skin-snehit.keras
32/32 ━━━━━━━━━━━━━━━━━━━━ 14s 425ms/step - accuracy: 0.8762 - loss:
0.3239 - val_accuracy: 0.6316 - val_loss: 0.9288 - learning_rate:
0.0010
Epoch 11/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 470ms/step - accuracy: 0.8462 - loss:
0.4219
Epoch 11: val_accuracy improved from 0.63158 to 0.64912, saving model
to skin-snehit.keras
32/32 ━━━━━━━━━━━━━━━━━━━━ 15s 485ms/step - accuracy: 0.8458 - loss:
0.4224 - val_accuracy: 0.6491 - val_loss: 1.0941 - learning_rate:
0.0010
Epoch 12/60
```

```
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 499ms/step - accuracy: 0.8599 - loss:
0.3262
Epoch 12: val_accuracy improved from 0.64912 to 0.70175, saving model
to skin-snehit.keras
32/32 ━━━━━━━━━━━━━━━━━━━━ 16s 514ms/step - accuracy: 0.8596 - loss:
0.3271 - val_accuracy: 0.7018 - val_loss: 0.8304 - learning_rate:
0.0010
Epoch 13/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 384ms/step - accuracy: 0.9099 - loss:
0.3019
Epoch 13: val_accuracy improved from 0.70175 to 0.75439, saving model
to skin-snehit.keras
32/32 ━━━━━━━━━━━━━━━━━━━━ 13s 399ms/step - accuracy: 0.9093 - loss:
0.3024 - val_accuracy: 0.7544 - val_loss: 0.8846 - learning_rate:
0.0010
Epoch 14/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 424ms/step - accuracy: 0.8880 - loss:
0.3031
Epoch 14: val_accuracy did not improve from 0.75439
32/32 ━━━━━━━━━━━━━━━━━━━━ 14s 437ms/step - accuracy: 0.8882 - loss:
0.3035 - val_accuracy: 0.6491 - val_loss: 1.1195 - learning_rate:
0.0010
Epoch 15/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 447ms/step - accuracy: 0.9303 - loss:
0.1947
Epoch 15: val_accuracy did not improve from 0.75439

Epoch 15: ReduceLROnPlateau reducing learning rate to
0.0003000000142492354.
32/32 ━━━━━━━━━━━━━━━━━━━━ 15s 463ms/step - accuracy: 0.9300 - loss:
0.1953 - val_accuracy: 0.6316 - val_loss: 1.3797 - learning_rate:
0.0010
Epoch 16/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 399ms/step - accuracy: 0.9538 - loss:
0.1346
Epoch 16: val_accuracy improved from 0.75439 to 0.77193, saving model
to skin-snehit.keras
32/32 ━━━━━━━━━━━━━━━━━━━━ 13s 416ms/step - accuracy: 0.9536 - loss:
0.1347 - val_accuracy: 0.7719 - val_loss: 0.7079 - learning_rate:
3.0000e-04
Epoch 17/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 406ms/step - accuracy: 0.9620 - loss:
0.1199
Epoch 17: val_accuracy did not improve from 0.77193
32/32 ━━━━━━━━━━━━━━━━━━━━ 13s 418ms/step - accuracy: 0.9618 - loss:
0.1214 - val_accuracy: 0.7018 - val_loss: 0.9736 - learning_rate:
3.0000e-04
Epoch 18/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 405ms/step - accuracy: 0.9674 - loss:
```

```
0.1123
Epoch 18: val_accuracy did not improve from 0.77193

Epoch 18: ReduceLROnPlateau reducing learning rate to
9.000000427477062e-05.
32/32 ━━━━━━━━━━━━━━━━━━━━ 14s 421ms/step - accuracy: 0.9672 - loss:
0.1126 - val_accuracy: 0.7368 - val_loss: 1.0070 - learning_rate:
3.0000e-04
Epoch 19/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 371ms/step - accuracy: 0.9749 - loss:
0.0937
Epoch 19: val_accuracy did not improve from 0.77193
32/32 ━━━━━━━━━━━━━━━━━━━━ 12s 383ms/step - accuracy: 0.9750 - loss:
0.0933 - val_accuracy: 0.7018 - val_loss: 0.9479 - learning_rate:
9.0000e-05
Epoch 20/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 372ms/step - accuracy: 0.9658 - loss:
0.0847
Epoch 20: val_accuracy did not improve from 0.77193

Epoch 20: ReduceLROnPlateau reducing learning rate to
2.700000040931627e-05.
32/32 ━━━━━━━━━━━━━━━━━━━━ 12s 384ms/step - accuracy: 0.9659 - loss:
0.0848 - val_accuracy: 0.7719 - val_loss: 0.8381 - learning_rate:
9.0000e-05
Epoch 21/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 376ms/step - accuracy: 0.9477 - loss:
0.1009
Epoch 21: val_accuracy did not improve from 0.77193
32/32 ━━━━━━━━━━━━━━━━━━━━ 13s 388ms/step - accuracy: 0.9482 - loss:
0.1004 - val_accuracy: 0.7719 - val_loss: 0.8423 - learning_rate:
2.7000e-05
Epoch 22/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 365ms/step - accuracy: 0.9633 - loss:
0.1001
Epoch 22: val_accuracy did not improve from 0.77193

Epoch 22: ReduceLROnPlateau reducing learning rate to
8.100000013655517e-06.
32/32 ━━━━━━━━━━━━━━━━━━━━ 12s 378ms/step - accuracy: 0.9632 - loss:
0.1001 - val_accuracy: 0.7719 - val_loss: 0.8439 - learning_rate:
2.7000e-05
Epoch 23/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 438ms/step - accuracy: 0.9608 - loss:
0.1041
Epoch 23: val_accuracy did not improve from 0.77193
32/32 ━━━━━━━━━━━━━━━━━━━━ 14s 450ms/step - accuracy: 0.9607 - loss:
0.1041 - val_accuracy: 0.7719 - val_loss: 0.8566 - learning_rate:
8.1000e-06
Epoch 24/60
```

```
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 396ms/step - accuracy: 0.9738 - loss:
0.0918
Epoch 24: val_accuracy did not improve from 0.77193

Epoch 24: ReduceLROnPlateau reducing learning rate to
2.429999949526973e-06.
32/32 ━━━━━━━━━━━━━━━━━━━━ 13s 408ms/step - accuracy: 0.9736 - loss:
0.0921 - val_accuracy: 0.7719 - val_loss: 0.8648 - learning_rate:
8.1000e-06
Epoch 25/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 380ms/step - accuracy: 0.9881 - loss:
0.0663
Epoch 25: val_accuracy did not improve from 0.77193
32/32 ━━━━━━━━━━━━━━━━━━━━ 13s 392ms/step - accuracy: 0.9876 - loss:
0.0675 - val_accuracy: 0.7719 - val_loss: 0.8722 - learning_rate:
2.4300e-06
Epoch 26/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 398ms/step - accuracy: 0.9596 - loss:
0.1021
Epoch 26: val_accuracy did not improve from 0.77193

Epoch 26: ReduceLROnPlateau reducing learning rate to
7.289999985005124e-07.
32/32 ━━━━━━━━━━━━━━━━━━━━ 13s 413ms/step - accuracy: 0.9596 - loss:
0.1020 - val_accuracy: 0.7719 - val_loss: 0.8774 - learning_rate:
2.4300e-06
Epoch 27/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 383ms/step - accuracy: 0.9568 - loss:
0.0932
Epoch 27: val_accuracy did not improve from 0.77193
32/32 ━━━━━━━━━━━━━━━━━━━━ 13s 395ms/step - accuracy: 0.9568 - loss:
0.0934 - val_accuracy: 0.7719 - val_loss: 0.8795 - learning_rate:
7.2900e-07
Epoch 28/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 386ms/step - accuracy: 0.9633 - loss:
0.1014
Epoch 28: val_accuracy did not improve from 0.77193

Epoch 28: ReduceLROnPlateau reducing learning rate to
2.1870000637136398e-07.
32/32 ━━━━━━━━━━━━━━━━━━━━ 13s 401ms/step - accuracy: 0.9631 - loss:
0.1023 - val_accuracy: 0.7719 - val_loss: 0.8820 - learning_rate:
7.2900e-07
Epoch 29/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 385ms/step - accuracy: 0.9852 - loss:
0.0673
Epoch 29: val_accuracy did not improve from 0.77193
32/32 ━━━━━━━━━━━━━━━━━━━━ 13s 397ms/step - accuracy: 0.9848 - loss:
0.0682 - val_accuracy: 0.7719 - val_loss: 0.8855 - learning_rate:
2.1870e-07
```

```
Epoch 30/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 390ms/step - accuracy: 0.9649 - loss:
0.0893
Epoch 30: val_accuracy did not improve from 0.77193

Epoch 30: ReduceLROnPlateau reducing learning rate to
6.561000276406048e-08.
32/32 ━━━━━━━━━━━━━━━━━━━━ 13s 403ms/step - accuracy: 0.9648 - loss:
0.0894 - val_accuracy: 0.7719 - val_loss: 0.8846 - learning_rate:
2.1870e-07
Epoch 31/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 410ms/step - accuracy: 0.9728 - loss:
0.0968
Epoch 31: val_accuracy did not improve from 0.77193
32/32 ━━━━━━━━━━━━━━━━━━━━ 14s 421ms/step - accuracy: 0.9727 - loss:
0.0968 - val_accuracy: 0.7719 - val_loss: 0.8854 - learning_rate:
6.5610e-08
Epoch 32/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 383ms/step - accuracy: 0.9547 - loss:
0.1474
Epoch 32: val_accuracy did not improve from 0.77193

Epoch 32: ReduceLROnPlateau reducing learning rate to
1.9683000829218145e-08.
32/32 ━━━━━━━━━━━━━━━━━━━━ 13s 395ms/step - accuracy: 0.9549 - loss:
0.1459 - val_accuracy: 0.7719 - val_loss: 0.8860 - learning_rate:
6.5610e-08
Epoch 33/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 385ms/step - accuracy: 0.9768 - loss:
0.0670
Epoch 33: val_accuracy did not improve from 0.77193
32/32 ━━━━━━━━━━━━━━━━━━━━ 13s 398ms/step - accuracy: 0.9765 - loss:
0.0674 - val_accuracy: 0.7719 - val_loss: 0.8872 - learning_rate:
1.9683e-08
Epoch 34/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 377ms/step - accuracy: 0.9762 - loss:
0.0742
Epoch 34: val_accuracy did not improve from 0.77193

Epoch 34: ReduceLROnPlateau reducing learning rate to
5.904900035602622e-09.
32/32 ━━━━━━━━━━━━━━━━━━━━ 13s 388ms/step - accuracy: 0.9761 - loss:
0.0741 - val_accuracy: 0.7719 - val_loss: 0.8887 - learning_rate:
1.9683e-08
Epoch 35/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 386ms/step - accuracy: 0.9587 - loss:
0.0992
Epoch 35: val_accuracy did not improve from 0.77193
32/32 ━━━━━━━━━━━━━━━━━━━━ 13s 399ms/step - accuracy: 0.9591 - loss:
0.0986 - val_accuracy: 0.7719 - val_loss: 0.8884 - learning_rate:
```

```
5.9049e-09
Epoch 36/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 381ms/step - accuracy: 0.9703 - loss:
0.0893
Epoch 36: val_accuracy did not improve from 0.77193

Epoch 36: ReduceLROnPlateau reducing learning rate to
1.7714700373261393e-09.
32/32 ━━━━━━━━━━━━━━━━━━━━ 13s 393ms/step - accuracy: 0.9705 - loss:
0.0892 - val_accuracy: 0.7719 - val_loss: 0.8879 - learning_rate:
5.9049e-09
Epoch 37/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 402ms/step - accuracy: 0.9703 - loss:
0.0980
Epoch 37: val_accuracy did not improve from 0.77193
32/32 ━━━━━━━━━━━━━━━━━━━━ 13s 414ms/step - accuracy: 0.9703 - loss:
0.0980 - val_accuracy: 0.7719 - val_loss: 0.8883 - learning_rate:
1.7715e-09
Epoch 38/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 364ms/step - accuracy: 0.9654 - loss:
0.0898
Epoch 38: val_accuracy did not improve from 0.77193

Epoch 38: ReduceLROnPlateau reducing learning rate to
5.314410245205181e-10.
32/32 ━━━━━━━━━━━━━━━━━━━━ 12s 377ms/step - accuracy: 0.9653 - loss:
0.0902 - val_accuracy: 0.7719 - val_loss: 0.8892 - learning_rate:
1.7715e-09
Epoch 39/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 368ms/step - accuracy: 0.9643 - loss:
0.0904
Epoch 39: val_accuracy did not improve from 0.77193
32/32 ━━━━━━━━━━━━━━━━━━━━ 12s 378ms/step - accuracy: 0.9642 - loss:
0.0908 - val_accuracy: 0.7719 - val_loss: 0.8874 - learning_rate:
5.3144e-10
Epoch 40/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 349ms/step - accuracy: 0.9615 - loss:
0.0782
Epoch 40: val_accuracy did not improve from 0.77193

Epoch 40: ReduceLROnPlateau reducing learning rate to
1.5943230069481729e-10.
32/32 ━━━━━━━━━━━━━━━━━━━━ 12s 360ms/step - accuracy: 0.9618 - loss:
0.0780 - val_accuracy: 0.7719 - val_loss: 0.8871 - learning_rate:
5.3144e-10
Epoch 41/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 349ms/step - accuracy: 0.9693 - loss:
0.0748
Epoch 41: val_accuracy did not improve from 0.77193
32/32 ━━━━━━━━━━━━━━━━━━━━ 12s 360ms/step - accuracy: 0.9692 - loss:
```

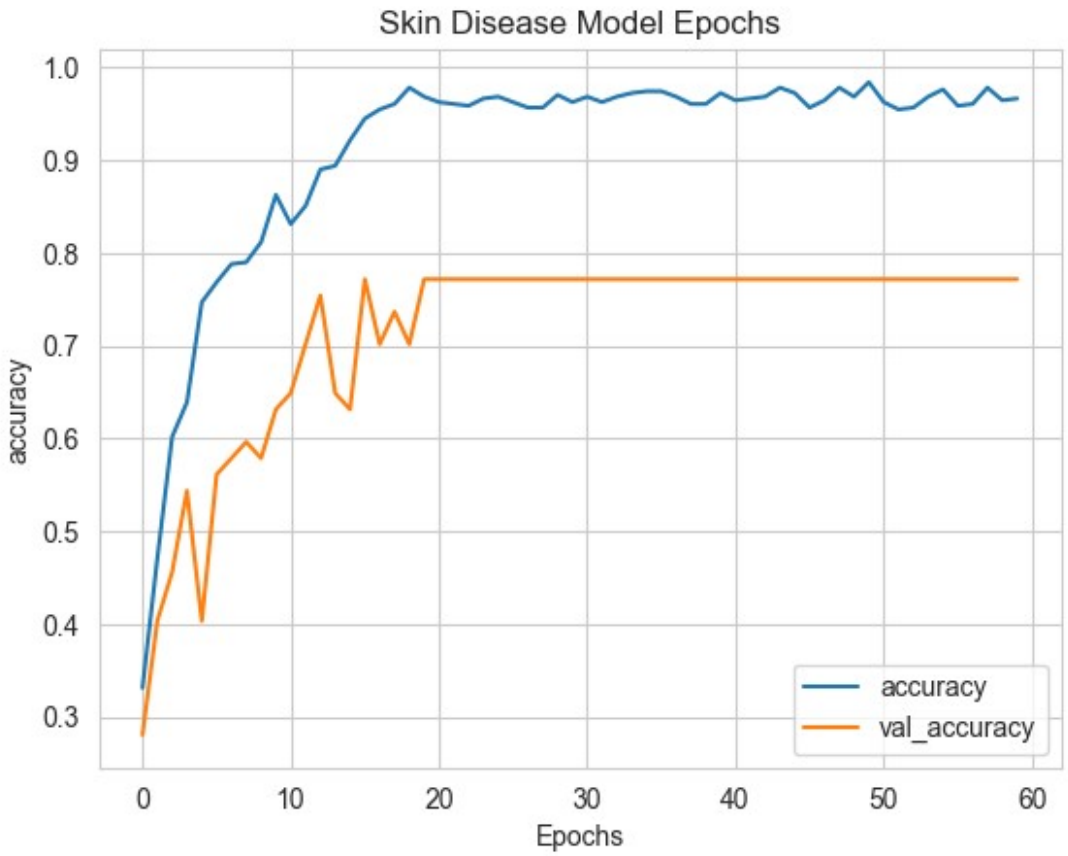
```
0.0752 - val_accuracy: 0.7719 - val_loss: 0.8862 - learning_rate:
1.5943e-10
Epoch 42/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 353ms/step - accuracy: 0.9776 - loss:
0.0739
Epoch 42: val_accuracy did not improve from 0.77193

Epoch 42: ReduceLROnPlateau reducing learning rate to
4.7829690208445185e-11.
32/32 ━━━━━━━━━━━━━━━━━━━━ 12s 365ms/step - accuracy: 0.9772 - loss:
0.0744 - val_accuracy: 0.7719 - val_loss: 0.8870 - learning_rate:
1.5943e-10
Epoch 43/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 350ms/step - accuracy: 0.9707 - loss:
0.0898
Epoch 43: val_accuracy did not improve from 0.77193
32/32 ━━━━━━━━━━━━━━━━━━━━ 12s 362ms/step - accuracy: 0.9707 - loss:
0.0899 - val_accuracy: 0.7719 - val_loss: 0.8868 - learning_rate:
4.7830e-11
Epoch 44/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 348ms/step - accuracy: 0.9739 - loss:
0.0652
Epoch 44: val_accuracy did not improve from 0.77193

Epoch 44: ReduceLROnPlateau reducing learning rate to
1.434890747886719e-11.
32/32 ━━━━━━━━━━━━━━━━━━━━ 12s 360ms/step - accuracy: 0.9741 - loss:
0.0653 - val_accuracy: 0.7719 - val_loss: 0.8878 - learning_rate:
4.7830e-11
Epoch 45/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 363ms/step - accuracy: 0.9719 - loss:
0.0827
Epoch 45: val_accuracy did not improve from 0.77193
32/32 ━━━━━━━━━━━━━━━━━━━━ 12s 374ms/step - accuracy: 0.9720 - loss:
0.0829 - val_accuracy: 0.7719 - val_loss: 0.8868 - learning_rate:
1.4349e-11
Epoch 46/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 343ms/step - accuracy: 0.9568 - loss:
0.1091
Epoch 46: val_accuracy did not improve from 0.77193

Epoch 46: ReduceLROnPlateau reducing learning rate to
4.304672243660157e-12.
32/32 ━━━━━━━━━━━━━━━━━━━━ 11s 355ms/step - accuracy: 0.9568 - loss:
0.1091 - val_accuracy: 0.7719 - val_loss: 0.8874 - learning_rate:
1.4349e-11
Epoch 47/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 351ms/step - accuracy: 0.9680 - loss:
0.0963
Epoch 47: val_accuracy did not improve from 0.77193
```

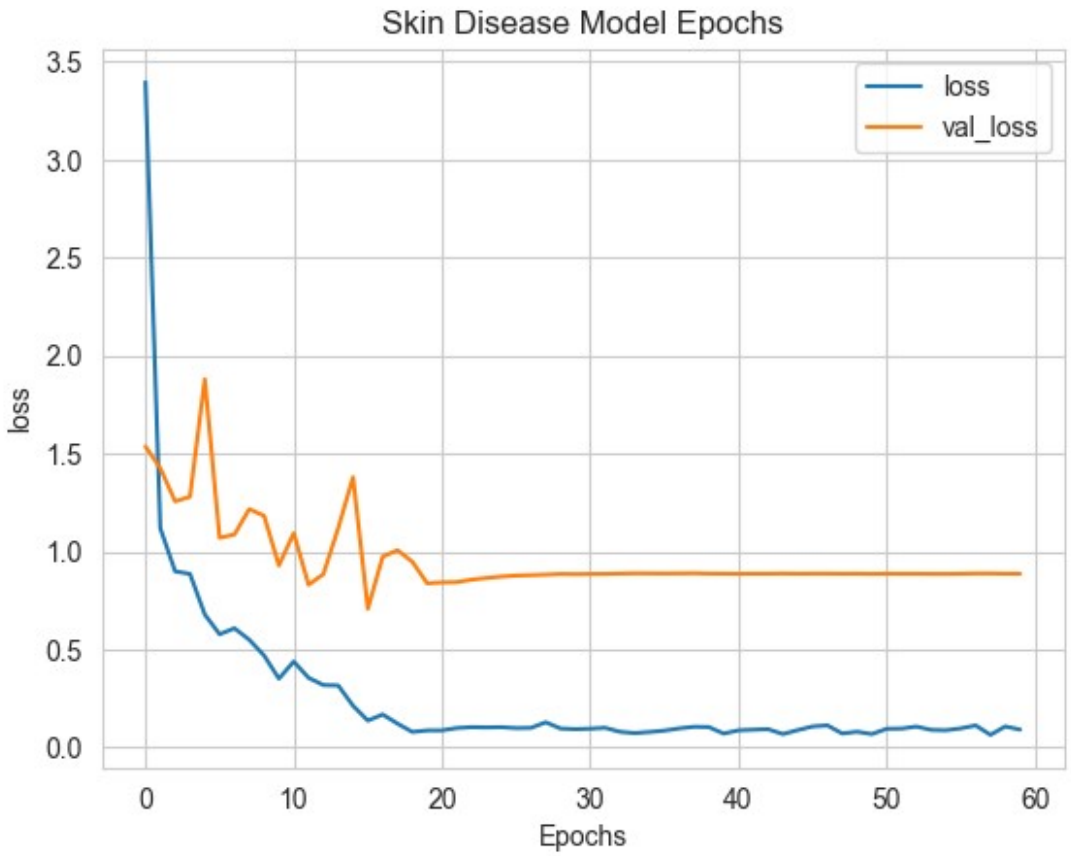
```
32/32 ━━━━━━━━━━━━━━━━━━━━ 12s 362ms/step - accuracy: 0.9679 - loss:
0.0968 - val_accuracy: 0.7719 - val_loss: 0.8874 - learning_rate:
4.3047e-12
Epoch 48/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 349ms/step - accuracy: 0.9782 - loss:
0.0768
Epoch 48: val_accuracy did not improve from 0.77193

Epoch 48: ReduceLROnPlateau reducing learning rate to
1.2914016210563428e-12.
32/32 ━━━━━━━━━━━━━━━━━━━━ 12s 362ms/step - accuracy: 0.9782 - loss:
0.0767 - val_accuracy: 0.7719 - val_loss: 0.8868 - learning_rate:
4.3047e-12
Epoch 49/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 353ms/step - accuracy: 0.9800 - loss:
0.0641
Epoch 49: val_accuracy did not improve from 0.77193
32/32 ━━━━━━━━━━━━━━━━━━━━ 12s 365ms/step - accuracy: 0.9796 - loss:
0.0646 - val_accuracy: 0.7719 - val_loss: 0.8870 - learning_rate:
1.2914e-12
Epoch 50/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 359ms/step - accuracy: 0.9847 - loss:
0.0588
Epoch 50: val_accuracy did not improve from 0.77193

Epoch 50: ReduceLROnPlateau reducing learning rate to
3.874204993273289e-13.
32/32 ━━━━━━━━━━━━━━━━━━━━ 12s 370ms/step - accuracy: 0.9847 - loss:
0.0591 - val_accuracy: 0.7719 - val_loss: 0.8859 - learning_rate:
1.2914e-12
Epoch 51/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 352ms/step - accuracy: 0.9614 - loss:
0.1033
Epoch 51: val_accuracy did not improve from 0.77193
32/32 ━━━━━━━━━━━━━━━━━━━━ 12s 362ms/step - accuracy: 0.9615 - loss:
0.1031 - val_accuracy: 0.7719 - val_loss: 0.8869 - learning_rate:
3.8742e-13
Epoch 52/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 353ms/step - accuracy: 0.9606 - loss:
0.0840
Epoch 52: val_accuracy did not improve from 0.77193

Epoch 52: ReduceLROnPlateau reducing learning rate to
1.162261530508052e-13.
32/32 ━━━━━━━━━━━━━━━━━━━━ 12s 366ms/step - accuracy: 0.9604 - loss:
0.0844 - val_accuracy: 0.7719 - val_loss: 0.8866 - learning_rate:
3.8742e-13
Epoch 53/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 353ms/step - accuracy: 0.9607 - loss:
0.0929
```

```
Epoch 53: val_accuracy did not improve from 0.77193
32/32 ━━━━━━━━━━━━━━━━━━━━ 12s 364ms/step - accuracy: 0.9606 - loss:
0.0933 - val_accuracy: 0.7719 - val_loss: 0.8866 - learning_rate:
1.1623e-13
Epoch 54/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 349ms/step - accuracy: 0.9672 - loss:
0.0856
Epoch 54: val_accuracy did not improve from 0.77193

Epoch 54: ReduceLROnPlateau reducing learning rate to
3.4867844288938296e-14.
32/32 ━━━━━━━━━━━━━━━━━━━━ 12s 361ms/step - accuracy: 0.9672 - loss:
0.0858 - val_accuracy: 0.7719 - val_loss: 0.8859 - learning_rate:
1.1623e-13
Epoch 55/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 351ms/step - accuracy: 0.9835 - loss:
0.0841
Epoch 55: val_accuracy did not improve from 0.77193
32/32 ━━━━━━━━━━━━━━━━━━━━ 12s 364ms/step - accuracy: 0.9832 - loss:
0.0842 - val_accuracy: 0.7719 - val_loss: 0.8858 - learning_rate:
3.4868e-14
Epoch 56/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 353ms/step - accuracy: 0.9567 - loss:
0.1012
Epoch 56: val_accuracy did not improve from 0.77193

Epoch 56: ReduceLROnPlateau reducing learning rate to
1.0460353083393582e-14.
32/32 ━━━━━━━━━━━━━━━━━━━━ 12s 365ms/step - accuracy: 0.9568 - loss:
0.1011 - val_accuracy: 0.7719 - val_loss: 0.8867 - learning_rate:
3.4868e-14
Epoch 57/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 354ms/step - accuracy: 0.9488 - loss:
0.1507
Epoch 57: val_accuracy did not improve from 0.77193
32/32 ━━━━━━━━━━━━━━━━━━━━ 12s 366ms/step - accuracy: 0.9491 - loss:
0.1495 - val_accuracy: 0.7719 - val_loss: 0.8881 - learning_rate:
1.0460e-14
Epoch 58/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 360ms/step - accuracy: 0.9810 - loss:
0.0628
Epoch 58: val_accuracy did not improve from 0.77193

Epoch 58: ReduceLROnPlateau reducing learning rate to
3.138105874196098e-15.
32/32 ━━━━━━━━━━━━━━━━━━━━ 12s 372ms/step - accuracy: 0.9809 - loss:
0.0629 - val_accuracy: 0.7719 - val_loss: 0.8882 - learning_rate:
1.0460e-14
Epoch 59/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 352ms/step - accuracy: 0.9642 - loss:
```

```
0.1022
Epoch 59: val_accuracy did not improve from 0.77193
32/32 ━━━━━━━━━━━━━━━━━━━━ 12s 364ms/step - accuracy: 0.9642 - loss:
0.1024 - val_accuracy: 0.7719 - val_loss: 0.8868 - learning_rate:
3.1381e-15
Epoch 60/60
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 356ms/step - accuracy: 0.9665 - loss:
0.0862
Epoch 60: val_accuracy did not improve from 0.77193

Epoch 60: ReduceLROnPlateau reducing learning rate to
9.414317622588293e-16.
32/32 ━━━━━━━━━━━━━━━━━━━━ 12s 368ms/step - accuracy: 0.9665 - loss:
0.0863 - val_accuracy: 0.7719 - val_loss: 0.8869 - learning_rate:
3.1381e-15
```

```python
#Visualize Training
def plot_graphs(history, string):
    sns.set_style("whitegrid")
    plt.plot(history.history[string])
    plt.plot(history.history["val_"+string])
    plt.xlabel("Epochs")
    plt.ylabel(string)
    plt.title("Skin Disease Model Epochs")
    plt.legend([string,"val_"+string])
    plt.show()
plot_graphs(history,'accuracy')
plot_graphs(history,'loss')
```


Skin Disease Model Epochs
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
accuracy
0
10
20
30
40
50
60
Epochs
accuracy
val_accuracy

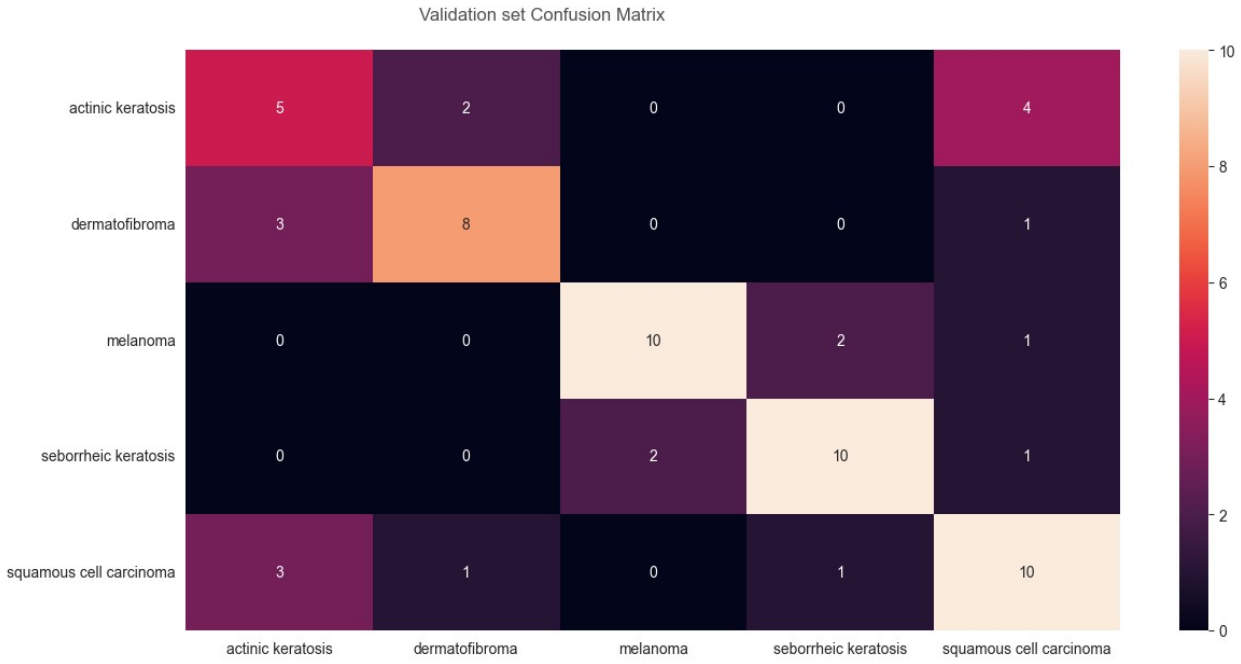



```
pred = model.predict(X_test)
pred = np.argmax(pred,axis=1)
y_test_new = np.argmax(y_test,axis=1)
print(classification_report(y_test_new,pred))
```

```
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 141ms/step
              precision    recall  f1-score   support

           0       0.45      0.45      0.45        11
           1       0.73      0.67      0.70        12
           2       0.83      0.77      0.80        13
           3       0.77      0.77      0.77        13
           4       0.59      0.67      0.62        15

    accuracy                           0.67        64
   macro avg       0.67      0.67      0.67        64
weighted avg       0.68      0.67      0.67        64
```

```
fig,ax=plt.subplots(1,1,figsize=(14,7))
sns.heatmap(confusion_matrix(y_test_new,pred),ax=ax,xticklabels=labels
,yticklabels=labels,annot=True)
fig.text(s='Validation set Confusion
Matrix',size=12,y=0.92,x=0.28,alpha=0.8)
```

```
plt.show()
```



Callbacks -> Callbacks can help you fix bugs more quickly, and can help you build better models. They can help you visualize how your model's training is going, and can even help prevent overfitting by implementing early stopping or customizing the learning rate on each iteration. By definition, "A callback is a set of functions to be applied at given stages of the training procedure. You can use callbacks to get a view on internal states and statistics of the model during training."

In this notebook, I'll be using **TensorBoard, ModelCheckpoint and ReduceLROnPlateau** callback functions

```
model.save('skin-snehit.keras')
```

Transfer Learning _ EfficientNet B0

Deep convolutional neural network models may take days or even weeks to train on very large datasets.

A way to short-cut this process is to re-use the model weights from pre-trained models that were developed for standard computer vision benchmark datasets, such as the ImageNet image recognition tasks. Top performing models can be downloaded and used directly, or integrated into a new model for your own computer vision problems.

In this notebook, I'll be using the **EfficientNetB0** model which will use the weights from the **ImageNet** dataset.

The include_top parameter is set to *False* so that the network doesn't include the top layer/ output layer from the pre-built model which allows us to add our own output layer depending upon our use case!

```
effnet =
EfficientNetB0(weights='imagenet',include_top=False,input_shape=(image
_size,image_size,3))
```

```
Downloading data from https://storage.googleapis.com/keras-
applications/efficientnetb0_notop.h5
16705208/16705208 ━━━━━━━━━━━━━━━━━━━━ 1s 0us/step
```

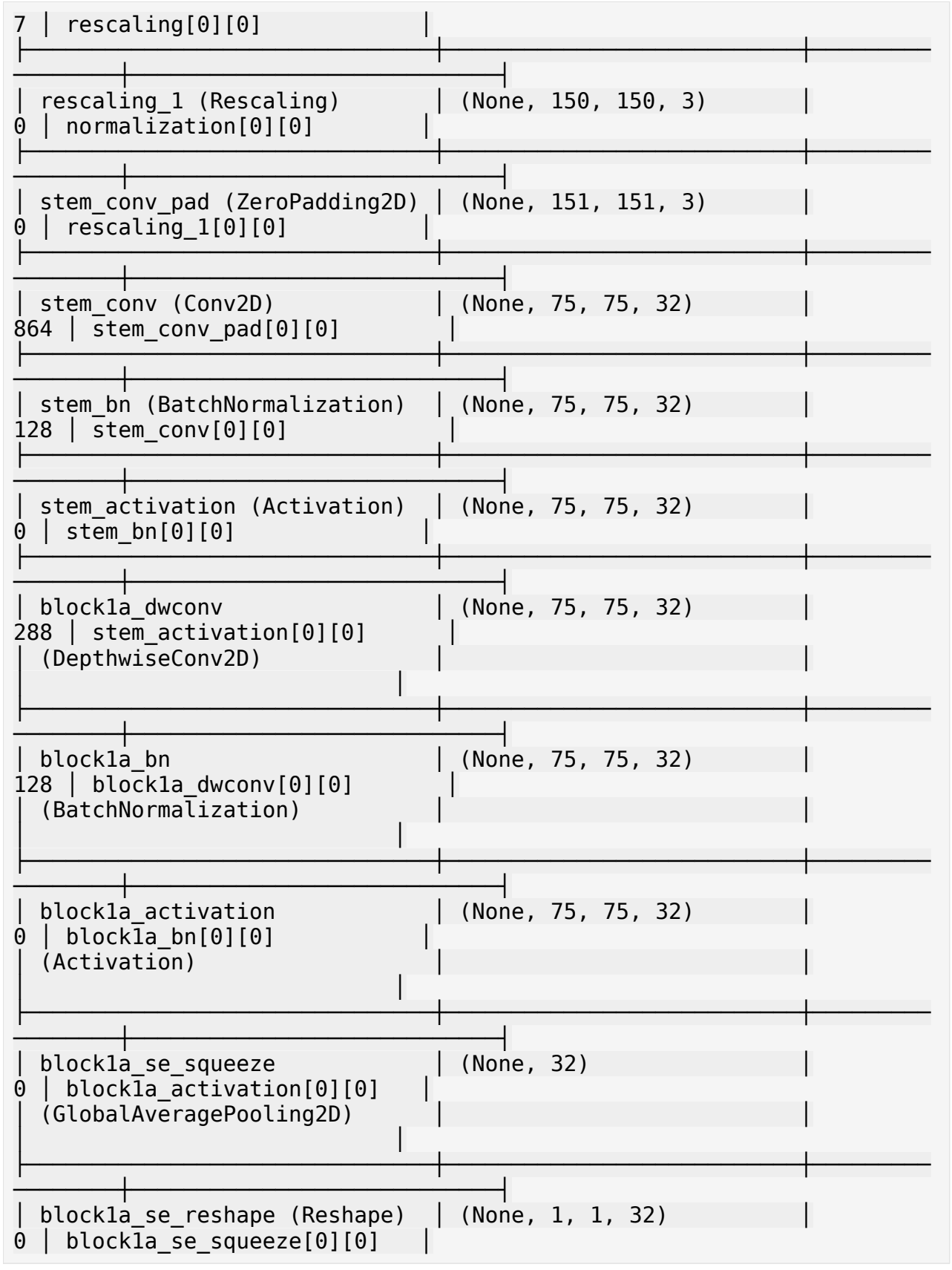
GlobalAveragePooling2D -> This layer acts similar to the Max Pooling layer in CNNs, the only difference being is that it uses the Average values instead of the Max value while *pooling*. This really helps in decreasing the computational load on the machine while training. **Dropout** -> This layer omits some of the neurons at each step from the layer making the neurons more independent from the neibouring neurons. It helps in avoiding overfitting. Neurons to be ommitted are selected at random. The **rate** parameter is the liklihood of a neuron activation being set to 0, thus dropping out the neuron

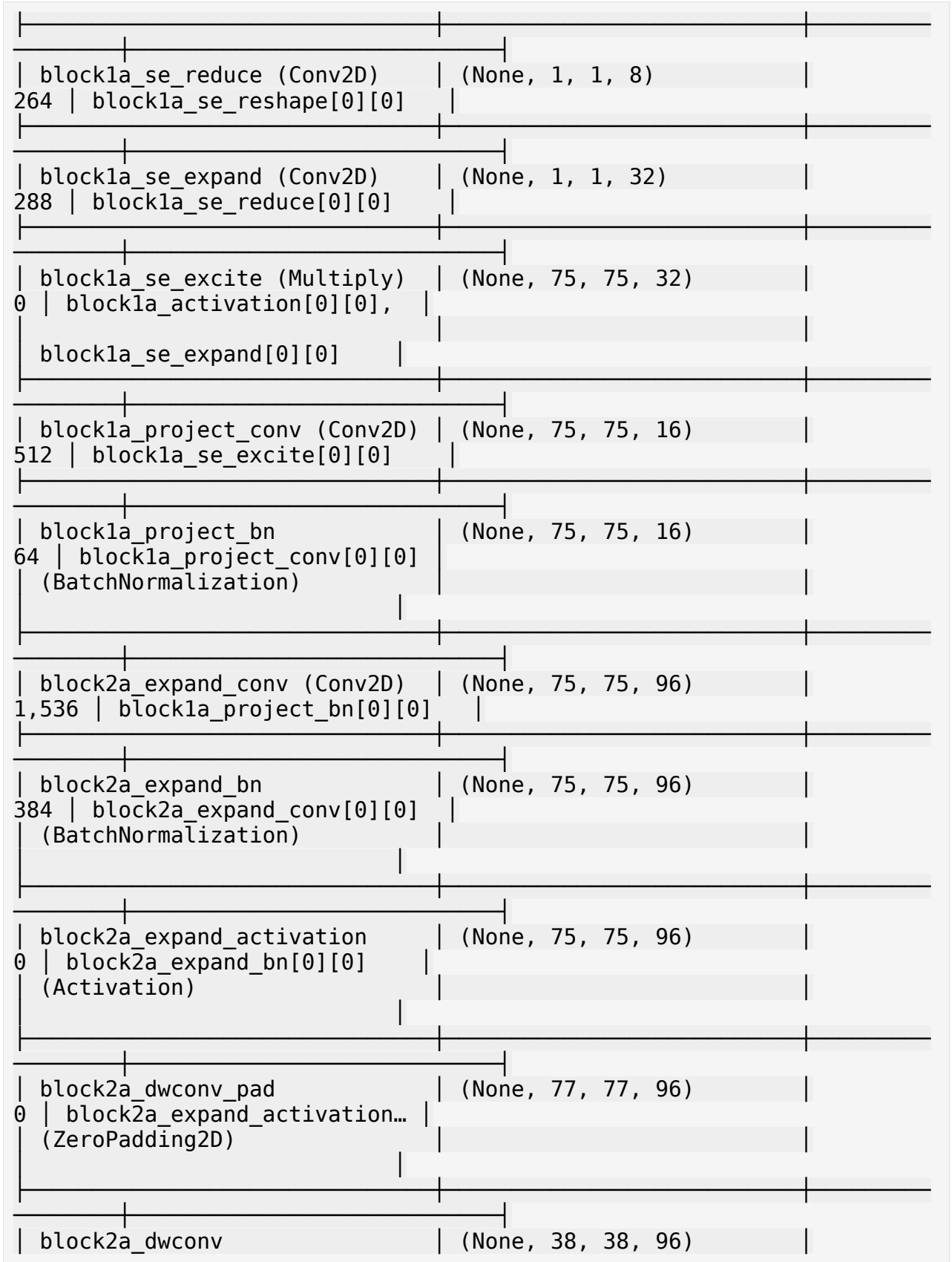
Dense -> This is the output layer which classifies the image into 1 of the 4 possible classes. It uses the **softmax** function which is a generalization of the sigmoid function.

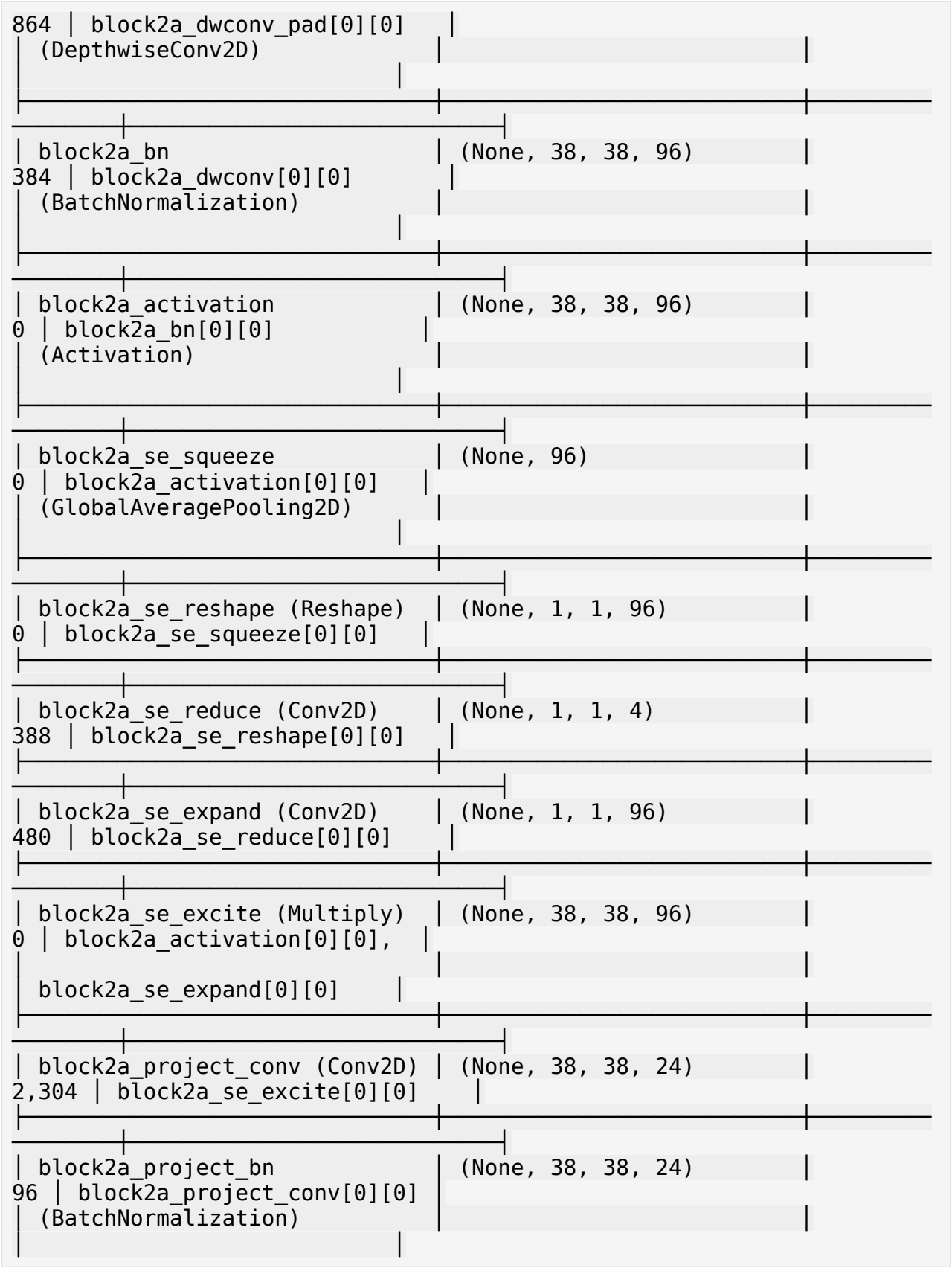
```
tf.keras.backend.clear_session()

model = effnet.output
model = tf.keras.layers.GlobalAveragePooling2D()(model)
model = tf.keras.layers.Dropout(rate=0.5)(model)
model = tf.keras.layers.Dense(5,activation='softmax')(model)
model = tf.keras.models.Model(inputs=effnet.input, outputs = model)

model.summary()
```

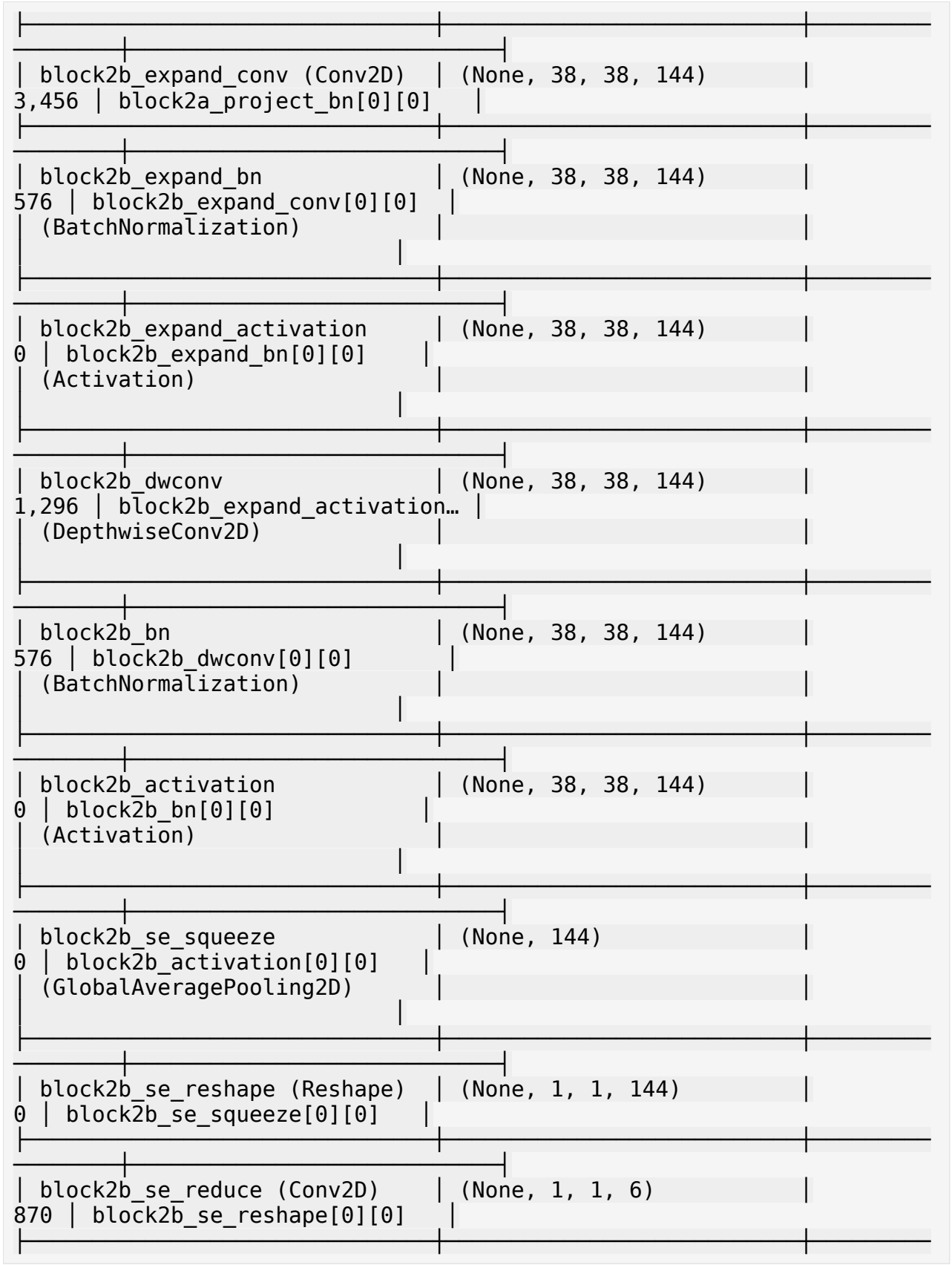
```
Model: "functional_1"
```

```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━
┓
┃ Layer (type)                    ┃ Output Shape           ┃
Param # ┃ Connected to               ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━
┩
│ input_layer_1 (InputLayer)      │ (None, 150, 150, 3)    │
0 │ -                              │
├─────────────────────────────────┼────────────────────────┼──────────────
──┤
│ rescaling (Rescaling)           │ (None, 150, 150, 3)    │
0 │ input_layer_1[0][0]            │
├─────────────────────────────────┼────────────────────────┼──────────────
──┤
│ normalization (Normalization)   │ (None, 150, 150, 3)    │
```

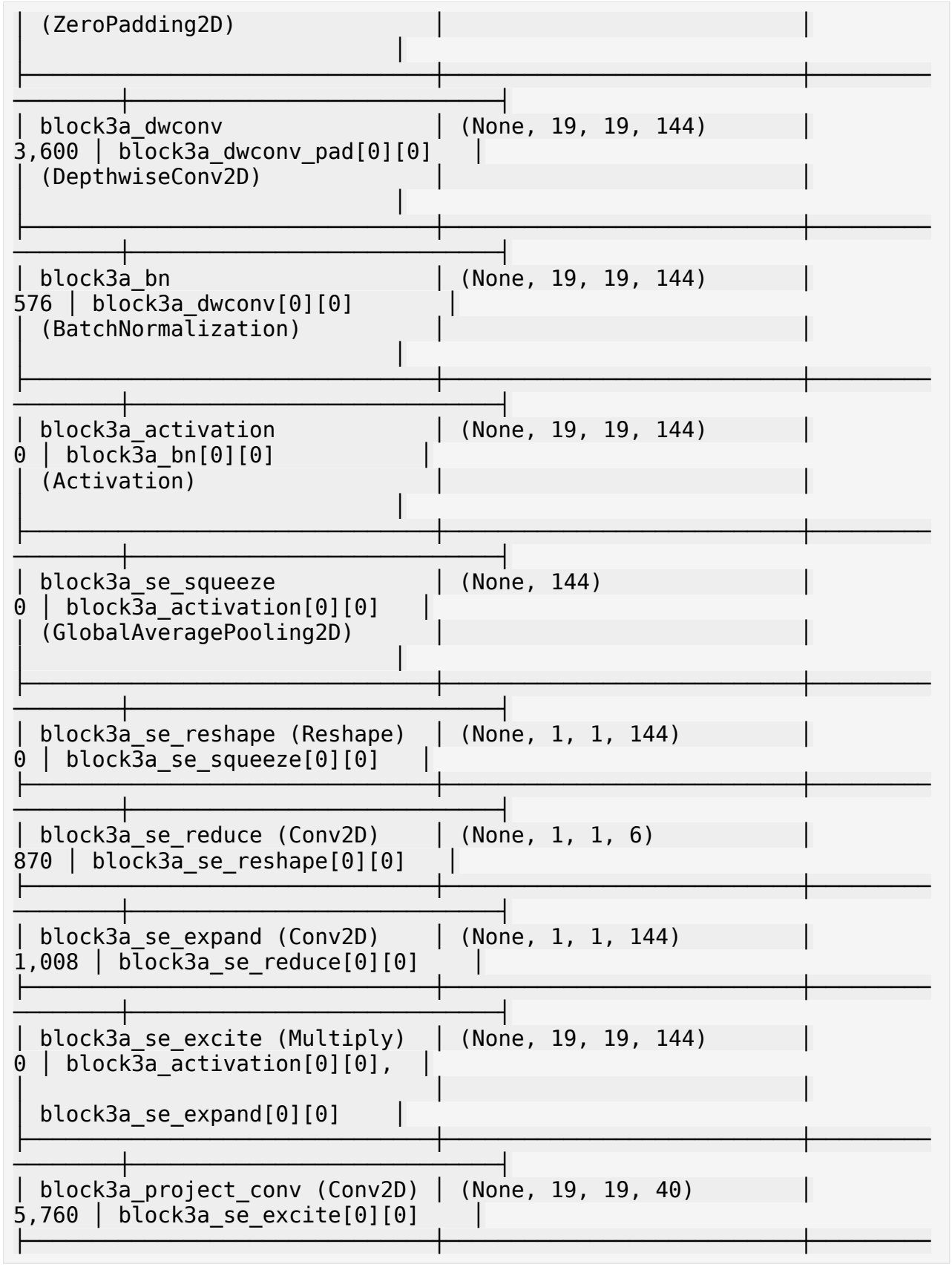

7 | rescaling[0][0]
rescaling_1 (Rescaling) | (None, 150, 150, 3) | 0 | normalization[0][0]
stem_conv_pad (ZeroPadding2D) | (None, 151, 151, 3) | 0 | rescaling_1[0][0]
stem_conv (Conv2D) | (None, 75, 75, 32) | 864 | stem_conv_pad[0][0]
stem_bn (BatchNormalization) | (None, 75, 75, 32) | 128 | stem_conv[0][0]
stem_activation (Activation) | (None, 75, 75, 32) | 0 | stem_bn[0][0]
block1a_dwconv (DepthwiseConv2D) | (None, 75, 75, 32) | 288 | stem_activation[0][0]
block1a_bn (BatchNormalization) | (None, 75, 75, 32) | 128 | block1a_dwconv[0][0]
block1a_activation (Activation) | (None, 75, 75, 32) | 0 | block1a_bn[0][0]
block1a_se_squeeze (GlobalAveragePooling2D) | (None, 32) | 0 | block1a_activation[0][0]
block1a_se_reshape (Reshape) | (None, 1, 1, 32) | 0 | block1a_se_squeeze[0][0]

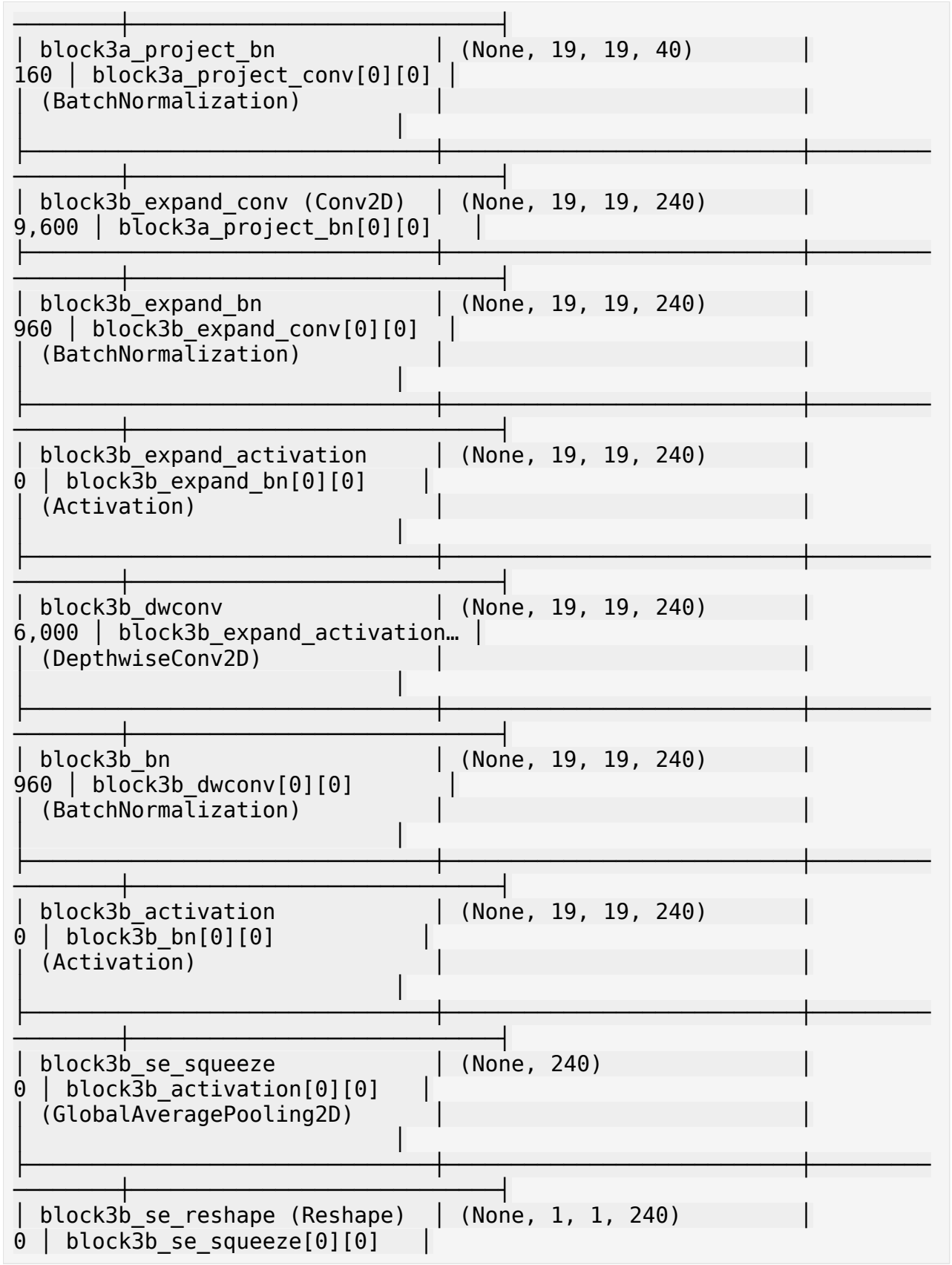

block1a_se_reduce (Conv2D) (None, 1, 1, 8) 264 block1a_se_reshape[0][0]
block1a_se_expand (Conv2D) (None, 1, 1, 32) 288 block1a_se_reduce[0][0]
block1a_se_excite (Multiply) (None, 75, 75, 32) 0 block1a_activation[0][0], block1a_se_expand[0][0]
block1a_project_conv (Conv2D) (None, 75, 75, 16) 512 block1a_se_excite[0][0]
block1a_project_bn (BatchNormalization) (None, 75, 75, 16) 64 block1a_project_conv[0][0]
block2a_expand_conv (Conv2D) (None, 75, 75, 96) 1,536 block1a_project_bn[0][0]
block2a_expand_bn (BatchNormalization) (None, 75, 75, 96) 384 block2a_expand_conv[0][0]
block2a_expand_activation (Activation) (None, 75, 75, 96) 0 block2a_expand_bn[0][0]
block2a_dwconv_pad (ZeroPadding2D) (None, 77, 77, 96) 0 block2a_expand_activation…
block2a_dwconv (None, 38, 38, 96)

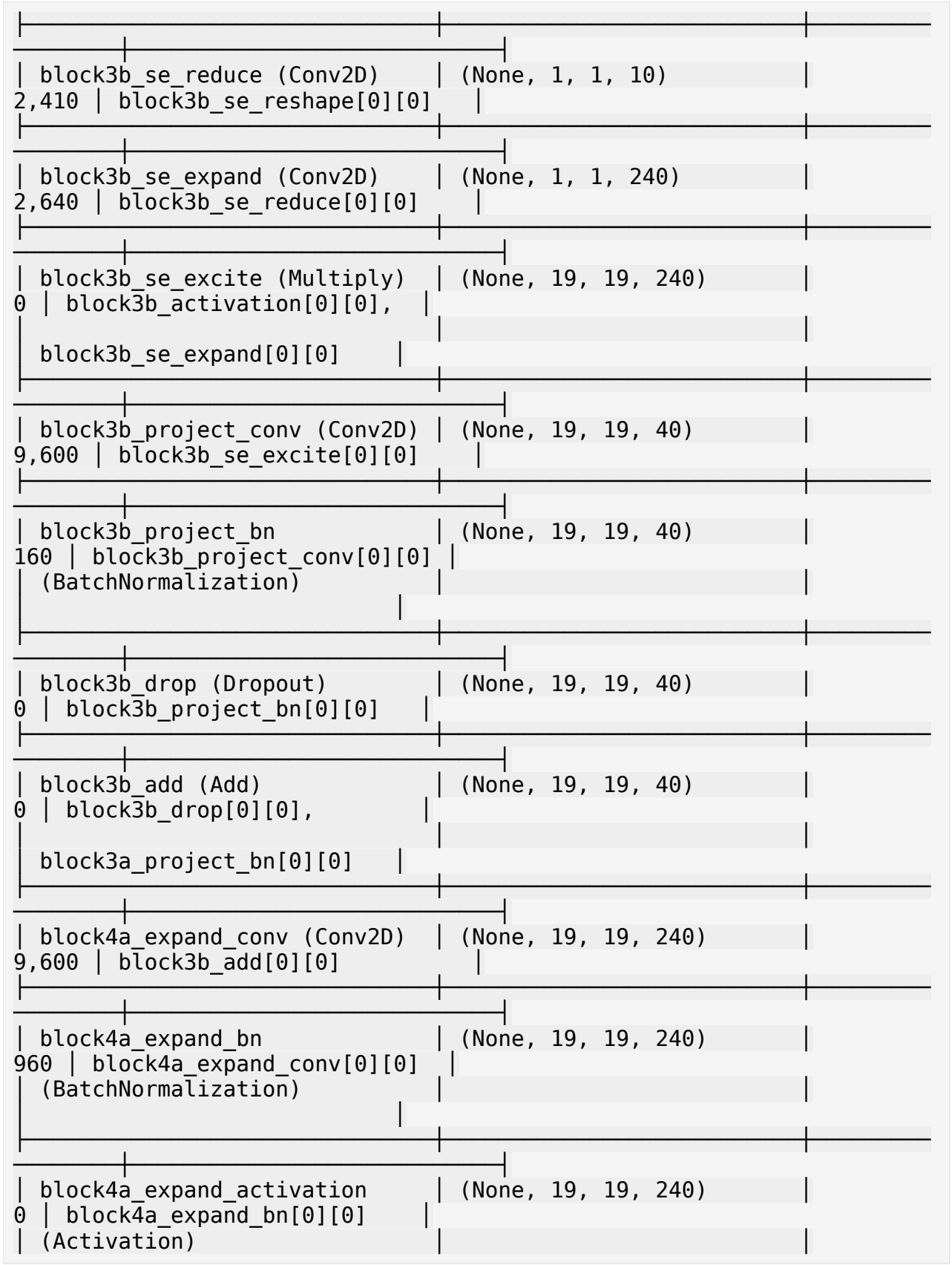

block2a_dwconv_pad[0][0] (DepthwiseConv2D) (None, 38, 38, 96) 864
block2a_bn (BatchNormalization) (None, 38, 38, 96) 384
block2a_activation (Activation) (None, 38, 38, 96) 0
block2a_se_squeeze (GlobalAveragePooling2D) (None, 96) 0
block2a_se_reshape (Reshape) (None, 1, 1, 96) 0
block2a_se_reduce (Conv2D) (None, 1, 1, 4) 388
block2a_se_expand (Conv2D) (None, 1, 1, 96) 480
block2a_se_excite (Multiply) (None, 38, 38, 96) 0
block2a_project_conv (Conv2D) (None, 38, 38, 24) 2,304
block2a_project_bn (BatchNormalization) (None, 38, 38, 24) 96

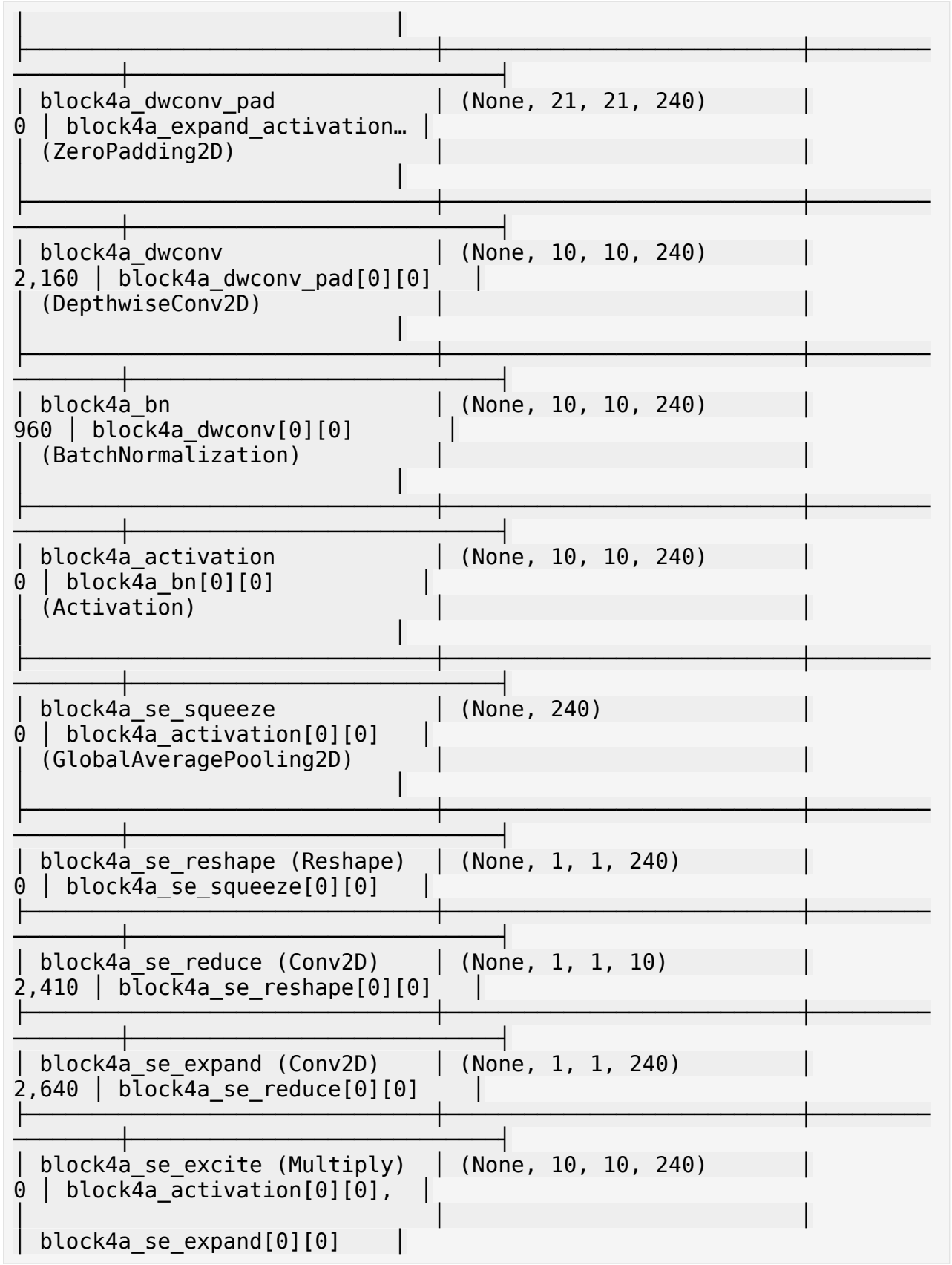


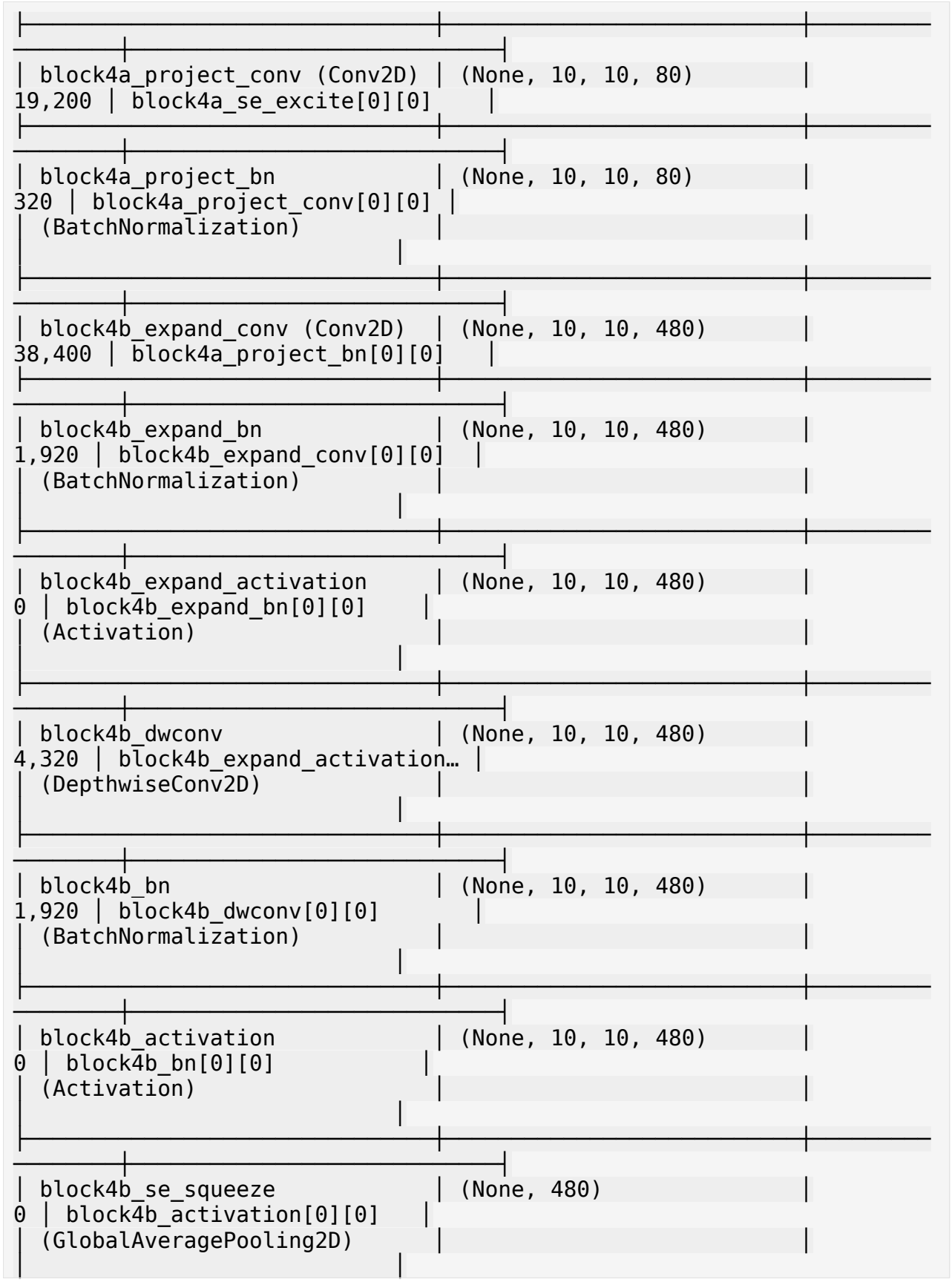
```
──────┼──────────────────────────────────┤
│ block2b_se_expand (Conv2D)    │ (None, 1, 1, 144)          │
1,008 │ block2b_se_reduce[0][0]    │
├──────────────────────────────┼──────────────────────────────┼──────
──────┼──────────────────────────────────┤
│ block2b_se_excite (Multiply)  │ (None, 38, 38, 144)        │
0 │ block2b_activation[0][0],  │
│                              │                              │
│ block2b_se_expand[0][0]    │
├──────────────────────────────┼──────────────────────────────┼──────
──────┼──────────────────────────────────┤
│ block2b_project_conv (Conv2D) │ (None, 38, 38, 24)         │
3,456 │ block2b_se_excite[0][0]    │
├──────────────────────────────┼──────────────────────────────┼──────
──────┼──────────────────────────────────┤
│ block2b_project_bn            │ (None, 38, 38, 24)         │
96 │ block2b_project_conv[0][0] │
│ (BatchNormalization)          │                              │
│                            │
├──────────────────────────────┼──────────────────────────────┼──────
──────┼──────────────────────────────────┤
│ block2b_drop (Dropout)        │ (None, 38, 38, 24)         │
0 │ block2b_project_bn[0][0]   │
├──────────────────────────────┼──────────────────────────────┼──────
──────┼──────────────────────────────────┤
│ block2b_add (Add)             │ (None, 38, 38, 24)         │
0 │ block2b_drop[0][0],        │
│                              │                              │
│ block2a_project_bn[0][0]   │
├──────────────────────────────┼──────────────────────────────┼──────
──────┼──────────────────────────────────┤
│ block3a_expand_conv (Conv2D)  │ (None, 38, 38, 144)        │
3,456 │ block2b_add[0][0]          │
├──────────────────────────────┼──────────────────────────────┼──────
──────┼──────────────────────────────────┤
│ block3a_expand_bn             │ (None, 38, 38, 144)        │
576 │ block3a_expand_conv[0][0]  │
│ (BatchNormalization)          │                              │
│                            │
├──────────────────────────────┼──────────────────────────────┼──────
──────┼──────────────────────────────────┤
│ block3a_expand_activation     │ (None, 38, 38, 144)        │
0 │ block3a_expand_bn[0][0]    │
│ (Activation)                  │                              │
│                            │
├──────────────────────────────┼──────────────────────────────┼──────
──────┼──────────────────────────────────┤
│ block3a_dwconv_pad            │ (None, 41, 41, 144)        │
0 │ block3a_expand_activation… │
```

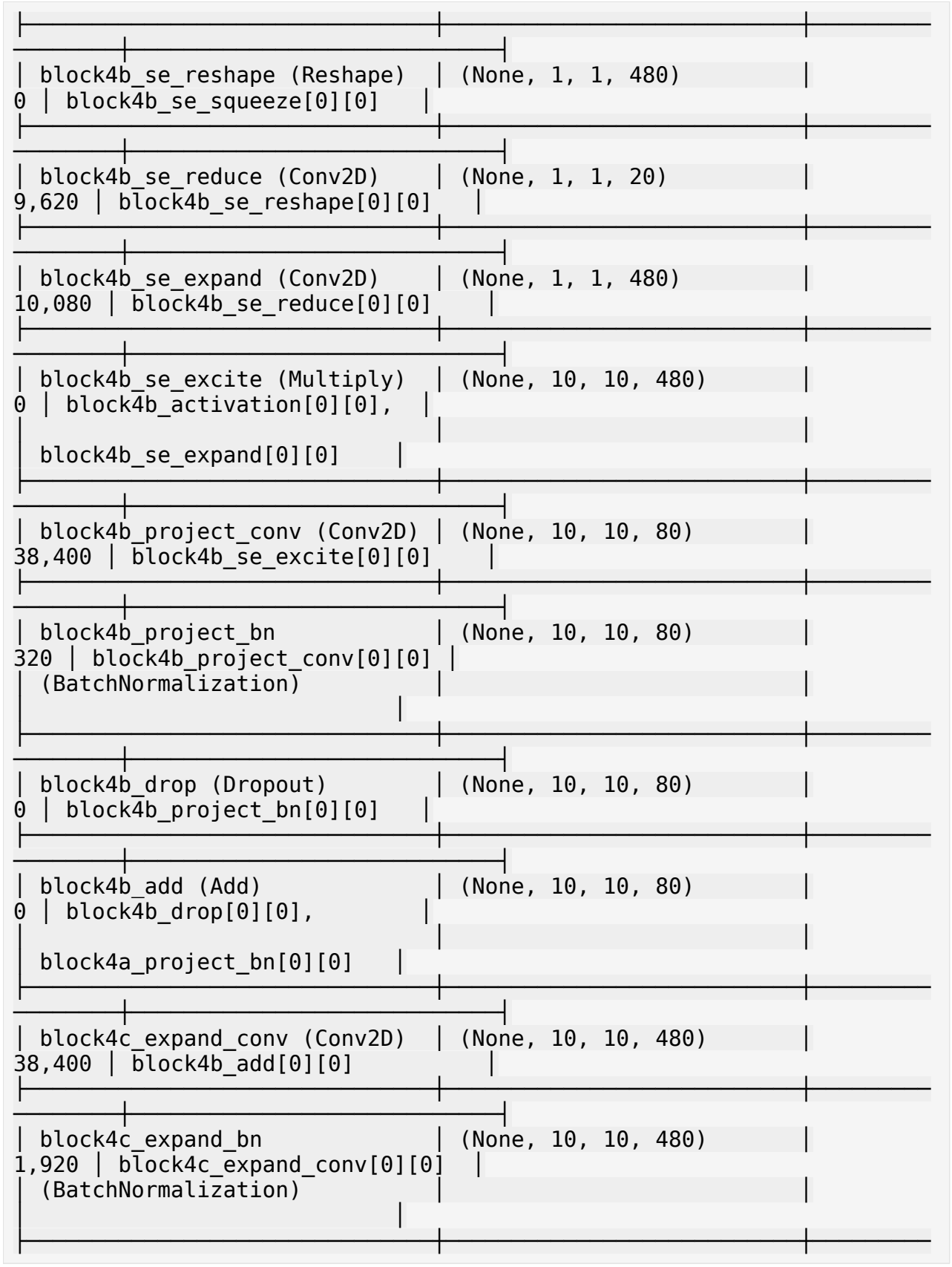


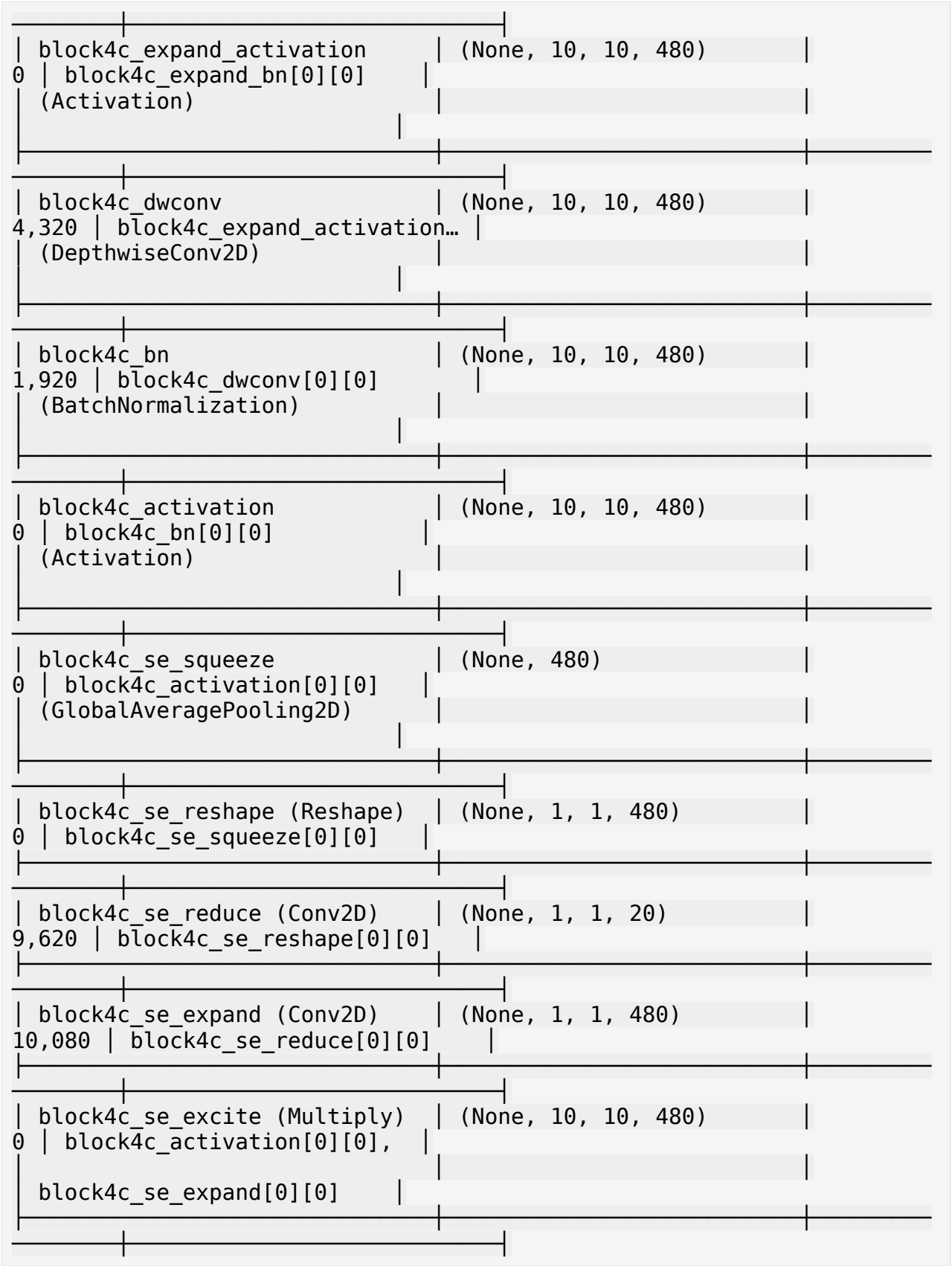




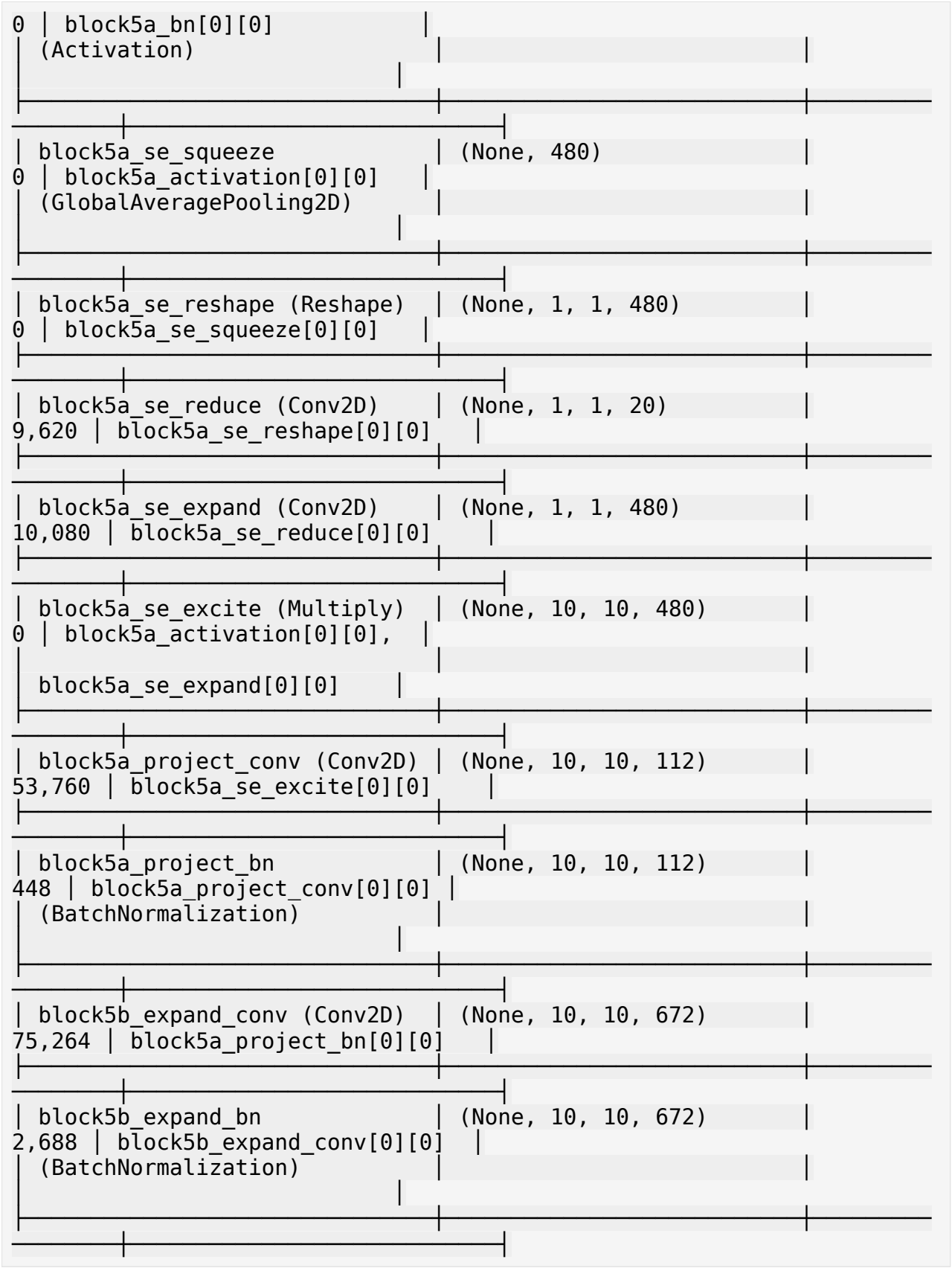

block4a_dwconv_pad (ZeroPadding2D) (None, 21, 21, 240) 0 block4a_expand_activation…
block4a_dwconv (DepthwiseConv2D) (None, 10, 10, 240) 2,160 block4a_dwconv_pad[0][0]
block4a_bn (BatchNormalization) (None, 10, 10, 240) 960 block4a_dwconv[0][0]
block4a_activation (Activation) (None, 10, 10, 240) 0 block4a_bn[0][0]
block4a_se_squeeze (GlobalAveragePooling2D) (None, 240) 0 block4a_activation[0][0]
block4a_se_reshape (Reshape) (None, 1, 1, 240) 0 block4a_se_squeeze[0][0]
block4a_se_reduce (Conv2D) (None, 1, 1, 10) 2,410 block4a_se_reshape[0][0]
block4a_se_expand (Conv2D) (None, 1, 1, 240) 2,640 block4a_se_reduce[0][0]
block4a_se_excite (Multiply) (None, 10, 10, 240) 0 block4a_activation[0][0], block4a_se_expand[0][0]

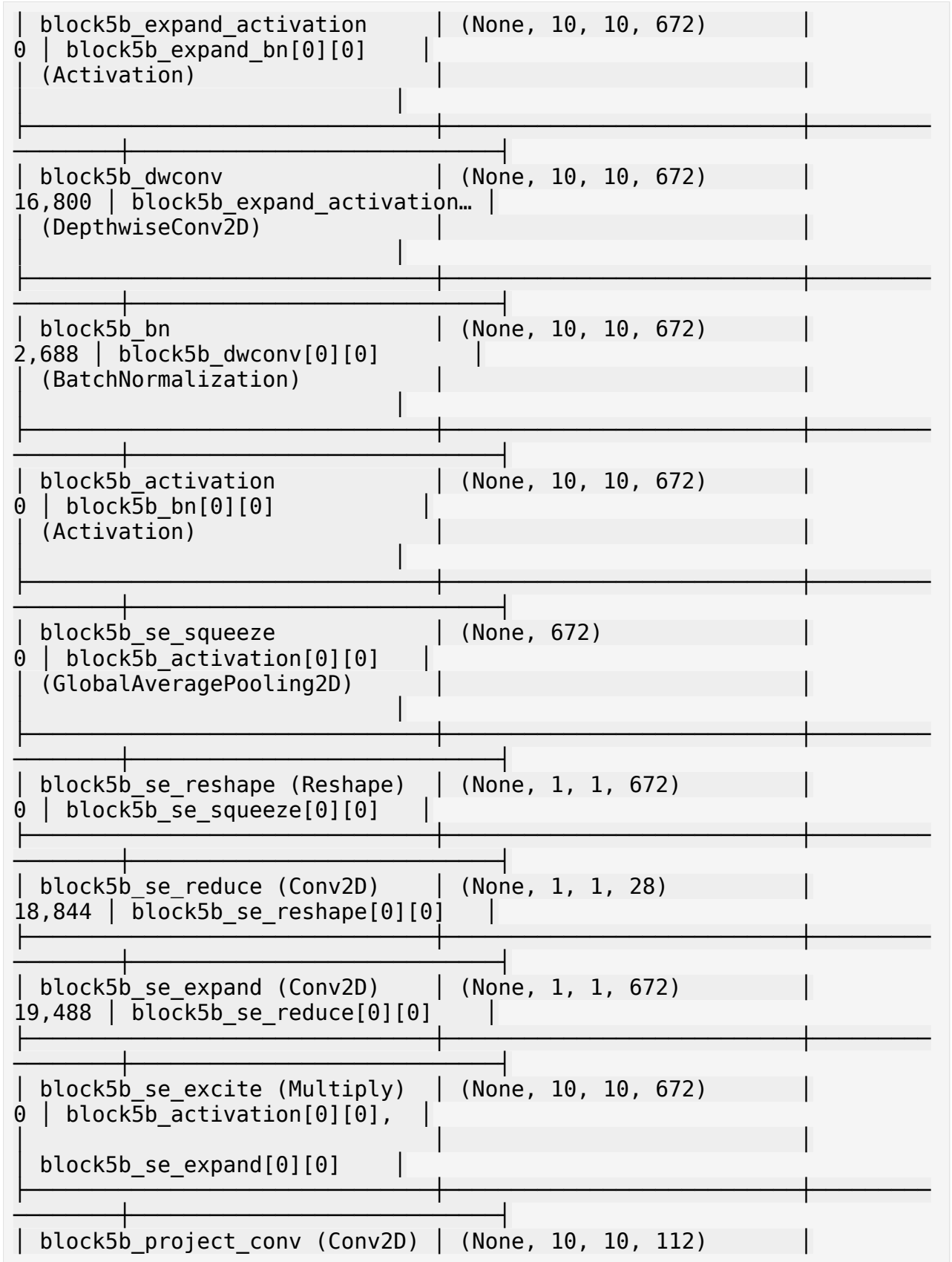

block4a_project_conv (Conv2D)	(None, 10, 10, 80)	19,200	block4a_se_excite[0][0]
block4a_project_bn (BatchNormalization)	(None, 10, 10, 80)	320	block4a_project_conv[0][0]
block4b_expand_conv (Conv2D)	(None, 10, 10, 480)	38,400	block4a_project_bn[0][0]
block4b_expand_bn (BatchNormalization)	(None, 10, 10, 480)	1,920	block4b_expand_conv[0][0]
block4b_expand_activation (Activation)	(None, 10, 10, 480)	0	block4b_expand_bn[0][0]
block4b_dwconv (DepthwiseConv2D)	(None, 10, 10, 480)	4,320	block4b_expand_activation…
block4b_bn (BatchNormalization)	(None, 10, 10, 480)	1,920	block4b_dwconv[0][0]
block4b_activation (Activation)	(None, 10, 10, 480)	0	block4b_bn[0][0]
block4b_se_squeeze (GlobalAveragePooling2D)	(None, 480)	0	block4b_activation[0][0]

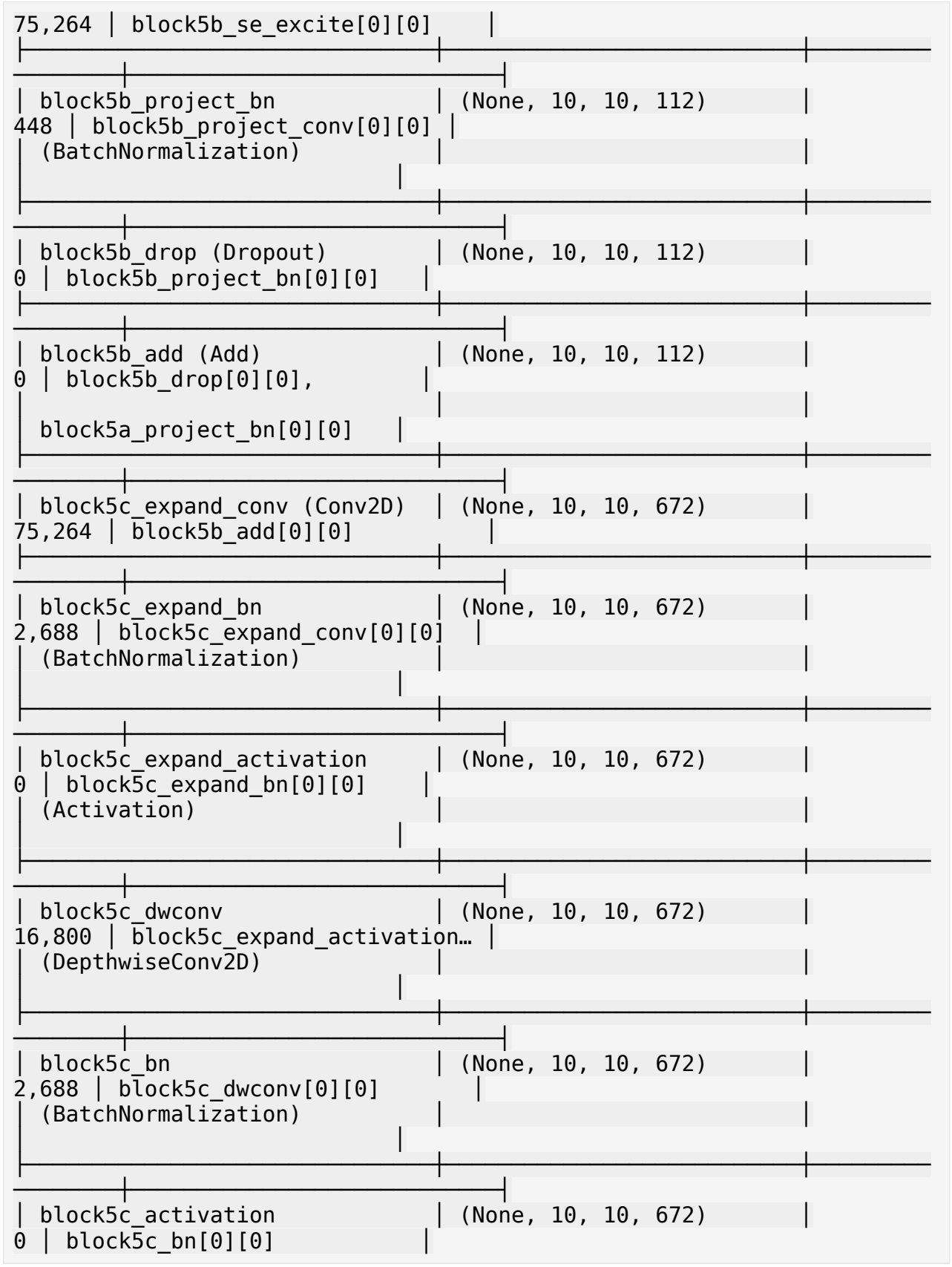

block4b_se_reshape (Reshape)	(None, 1, 1, 480)	0	block4b_se_squeeze[0][0]
block4b_se_reduce (Conv2D)	(None, 1, 1, 20)	9,620	block4b_se_reshape[0][0]
block4b_se_expand (Conv2D)	(None, 1, 1, 480)	10,080	block4b_se_reduce[0][0]
block4b_se_excite (Multiply)	(None, 10, 10, 480)	0	block4b_activation[0][0], block4b_se_expand[0][0]
block4b_project_conv (Conv2D)	(None, 10, 10, 80)	38,400	block4b_se_excite[0][0]
block4b_project_bn (BatchNormalization)	(None, 10, 10, 80)	320	block4b_project_conv[0][0]
block4b_drop (Dropout)	(None, 10, 10, 80)	0	block4b_project_bn[0][0]
block4b_add (Add)	(None, 10, 10, 80)	0	block4b_drop[0][0], block4a_project_bn[0][0]
block4c_expand_conv (Conv2D)	(None, 10, 10, 480)	38,400	block4b_add[0][0]
block4c_expand_bn (BatchNormalization)	(None, 10, 10, 480)	1,920	block4c_expand_conv[0][0]

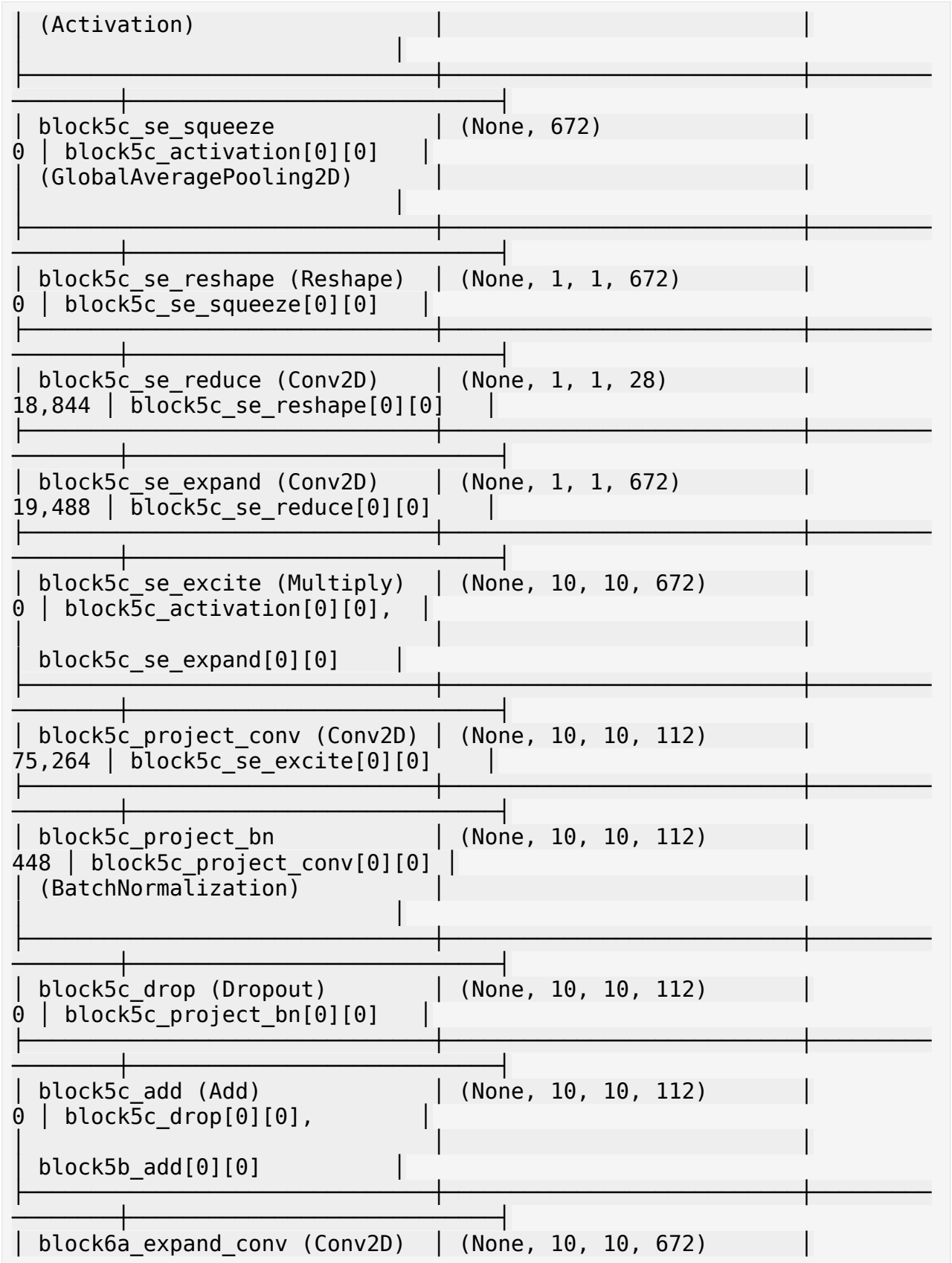


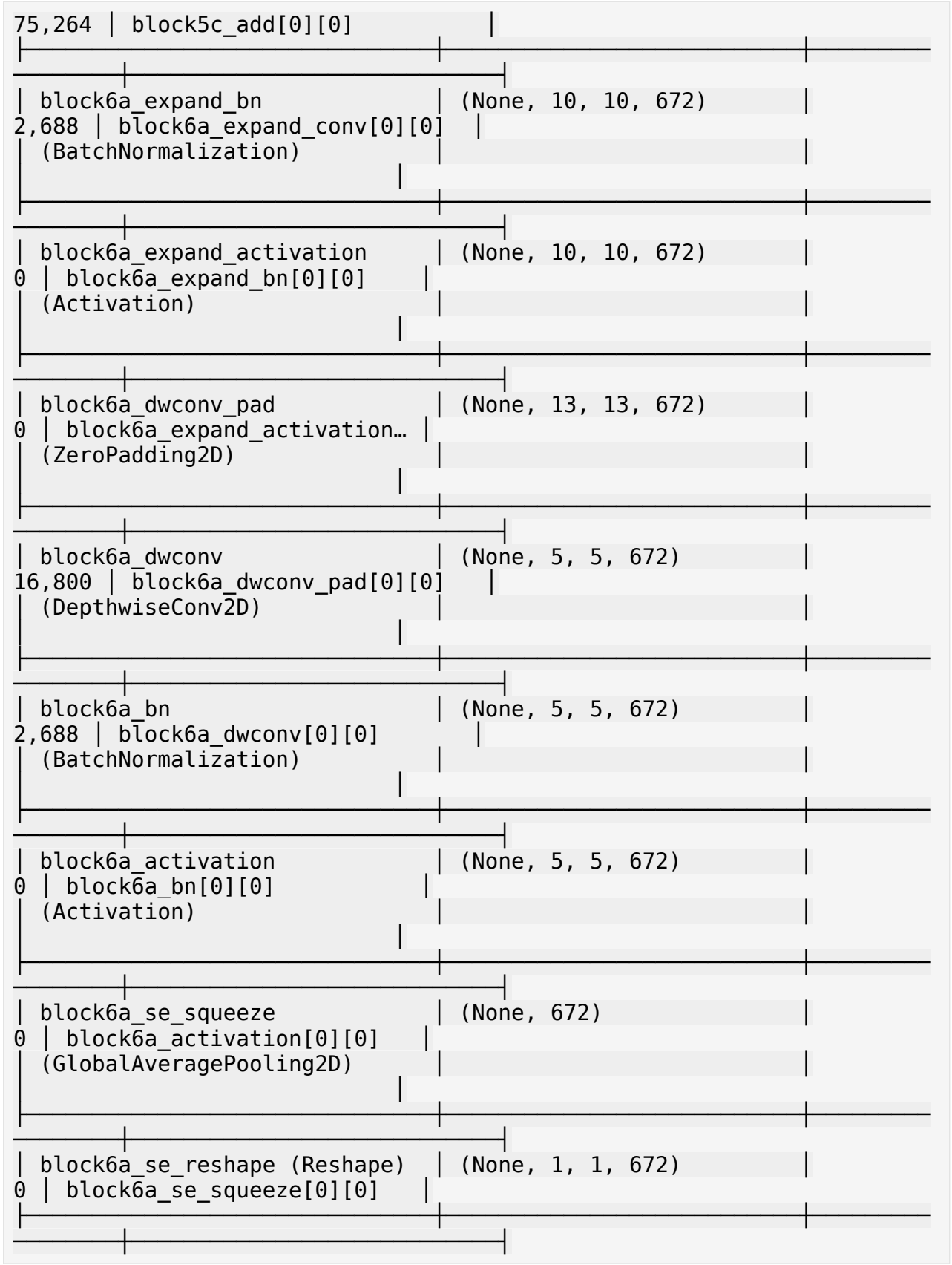
```
| block4c_project_conv (Conv2D) | (None, 10, 10, 80)         |
38,400 | block4c_se_excite[0][0]    |
├──────────────────────────────┼────────────────────────────┼──────────
──────┼──────────────────────────────┤
| block4c_project_bn            | (None, 10, 10, 80)         |
320 | block4c_project_conv[0][0] |
| (BatchNormalization)          |                            |
|                               |
├──────────────────────────────┼────────────────────────────┼──────────
──────┼──────────────────────────────┤
| block4c_drop (Dropout)        | (None, 10, 10, 80)         |
0 | block4c_project_bn[0][0]   |
├──────────────────────────────┼────────────────────────────┼──────────
──────┼──────────────────────────────┤
| block4c_add (Add)             | (None, 10, 10, 80)         |
0 | block4c_drop[0][0],        |
|                               |                            |
| block4b_add[0][0]             |
├──────────────────────────────┼────────────────────────────┼──────────
──────┼──────────────────────────────┤
| block5a_expand_conv (Conv2D)  | (None, 10, 10, 480)        |
38,400 | block4c_add[0][0]          |
├──────────────────────────────┼────────────────────────────┼──────────
──────┼──────────────────────────────┤
| block5a_expand_bn             | (None, 10, 10, 480)        |
1,920 | block5a_expand_conv[0][0]  |
| (BatchNormalization)          |                            |
|                               |
├──────────────────────────────┼────────────────────────────┼──────────
──────┼──────────────────────────────┤
| block5a_expand_activation     | (None, 10, 10, 480)        |
0 | block5a_expand_bn[0][0]    |
| (Activation)                  |                            |
|                               |
├──────────────────────────────┼────────────────────────────┼──────────
──────┼──────────────────────────────┤
| block5a_dwconv                | (None, 10, 10, 480)        |
12,000 | block5a_expand_activation… |
| (DepthwiseConv2D)             |                            |
|                               |
├──────────────────────────────┼────────────────────────────┼──────────
──────┼──────────────────────────────┤
| block5a_bn                    | (None, 10, 10, 480)        |
1,920 | block5a_dwconv[0][0]       |
| (BatchNormalization)          |                            |
|                               |
├──────────────────────────────┼────────────────────────────┼──────────
──────┼──────────────────────────────┤
| block5a_activation            | (None, 10, 10, 480)        |
```

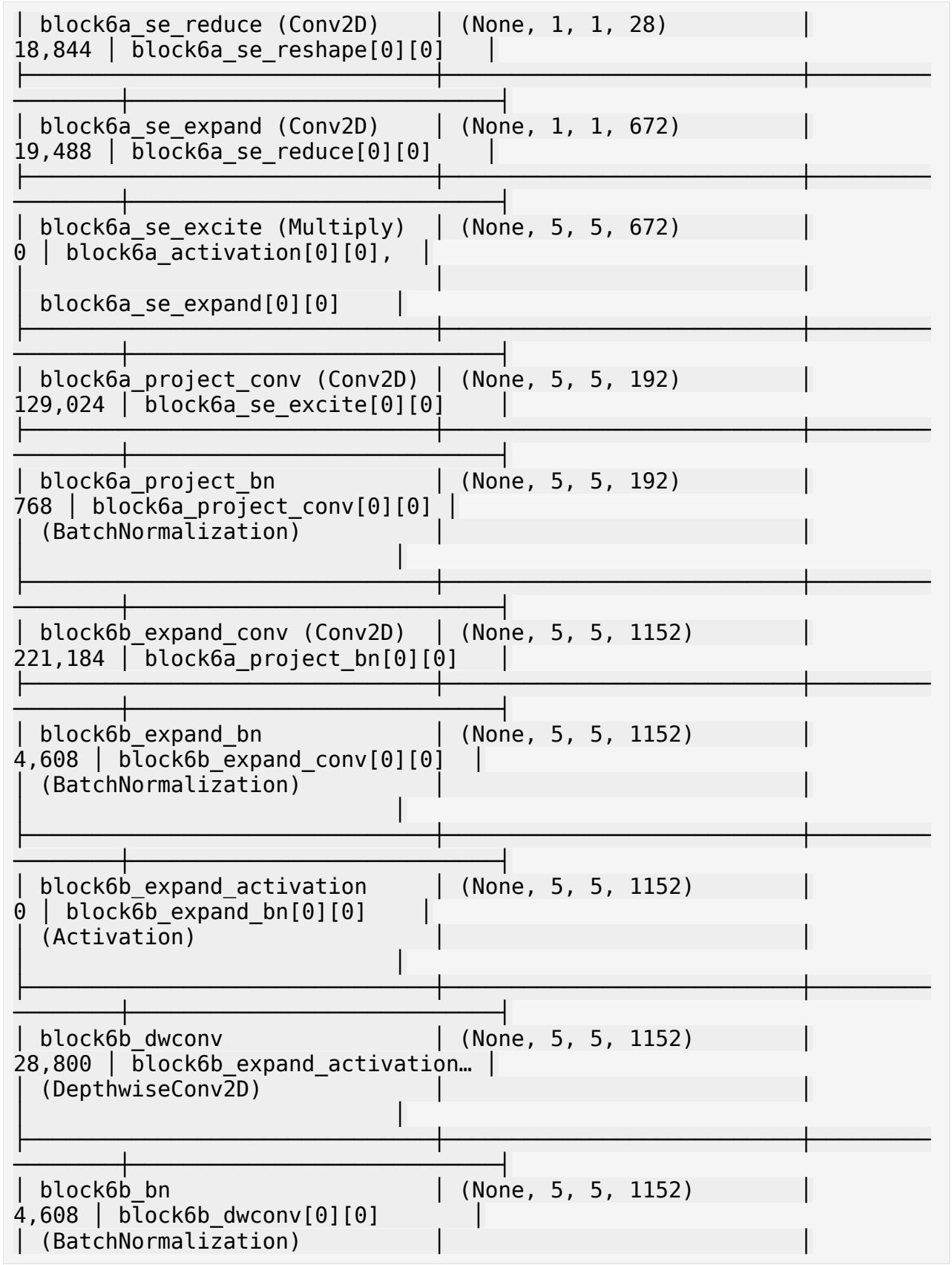


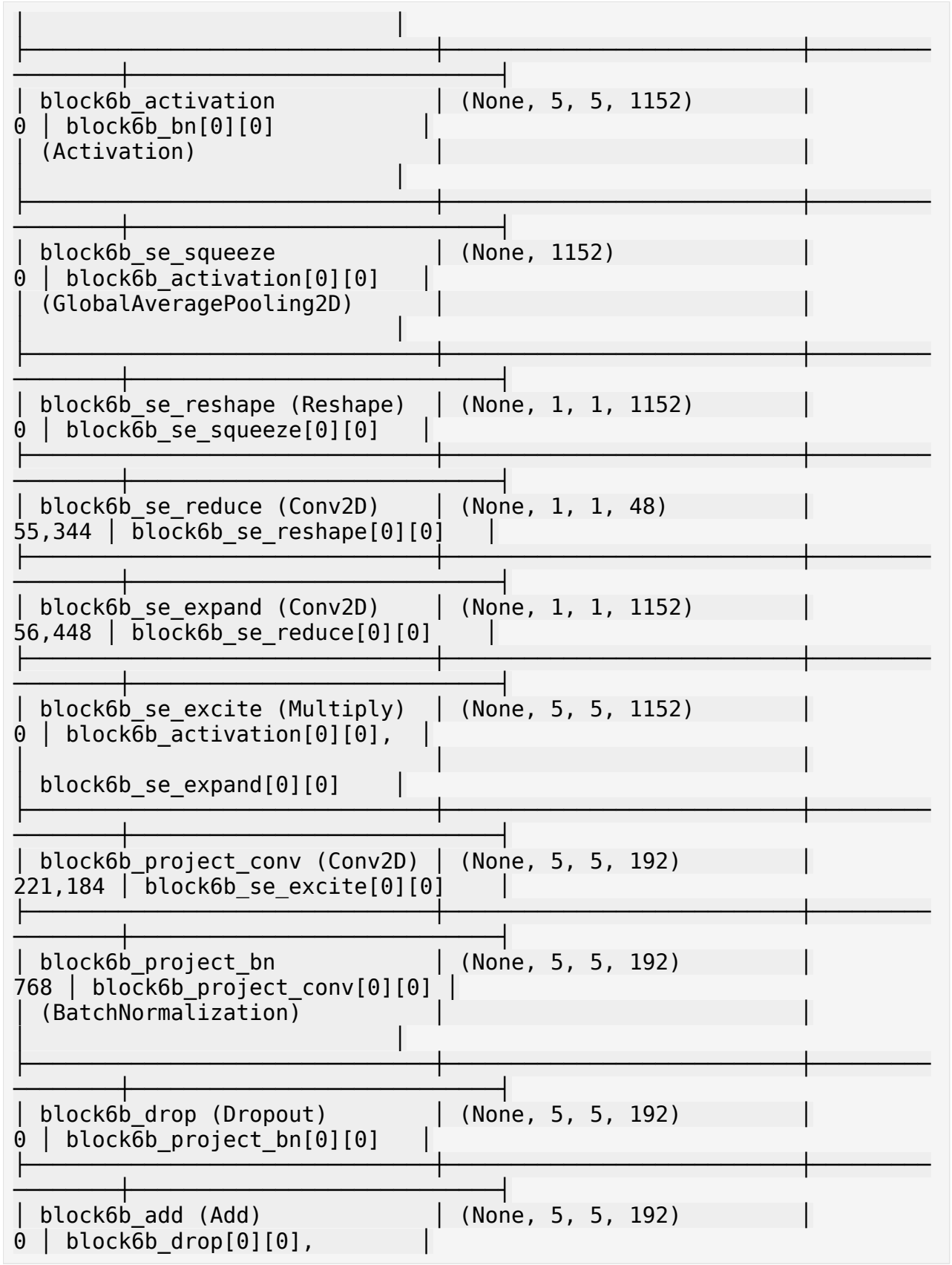

block5b_expand_activation (Activation) (None, 10, 10, 672) 0 block5b_expand_bn[0][0]
block5b_dwconv (DepthwiseConv2D) (None, 10, 10, 672) 16,800 block5b_expand_activation…
block5b_bn (BatchNormalization) (None, 10, 10, 672) 2,688 block5b_dwconv[0][0]
block5b_activation (Activation) (None, 10, 10, 672) 0 block5b_bn[0][0]
block5b_se_squeeze (GlobalAveragePooling2D) (None, 672) 0 block5b_activation[0][0]
block5b_se_reshape (Reshape) (None, 1, 1, 672) 0 block5b_se_squeeze[0][0]
block5b_se_reduce (Conv2D) (None, 1, 1, 28) 18,844 block5b_se_reshape[0][0]
block5b_se_expand (Conv2D) (None, 1, 1, 672) 19,488 block5b_se_reduce[0][0]
block5b_se_excite (Multiply) (None, 10, 10, 672) 0 block5b_activation[0][0], block5b_se_expand[0][0]
block5b_project_conv (Conv2D) (None, 10, 10, 112)

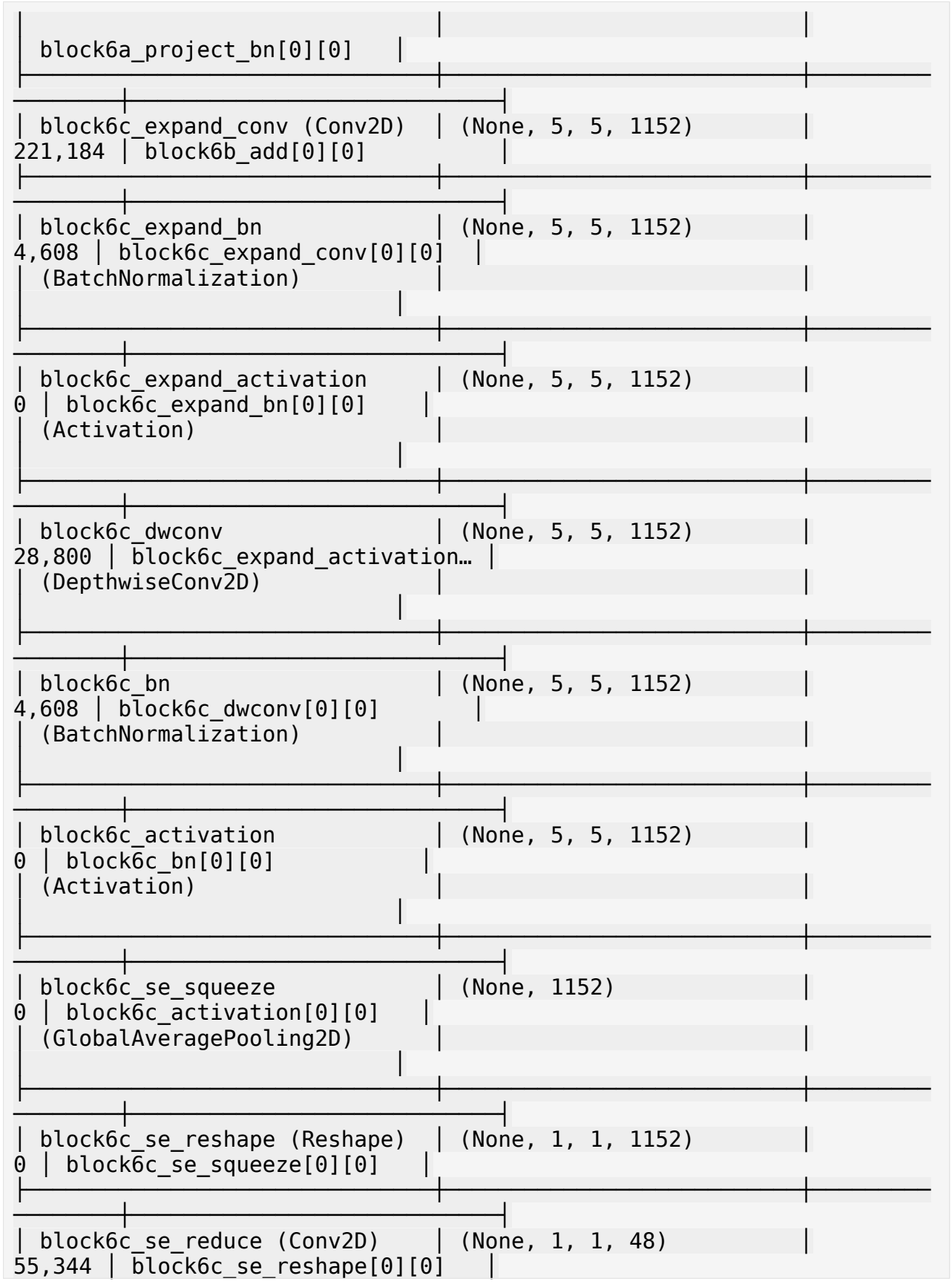

75,264 | block5b_se_excite[0][0]
block5b_project_bn (BatchNormalization) | (None, 10, 10, 112) | 448 | block5b_project_conv[0][0]
block5b_drop (Dropout) | (None, 10, 10, 112) | 0 | block5b_project_bn[0][0]
block5b_add (Add) | (None, 10, 10, 112) | 0 | block5b_drop[0][0], block5a_project_bn[0][0]
block5c_expand_conv (Conv2D) | (None, 10, 10, 672) | 75,264 | block5b_add[0][0]
block5c_expand_bn (BatchNormalization) | (None, 10, 10, 672) | 2,688 | block5c_expand_conv[0][0]
block5c_expand_activation (Activation) | (None, 10, 10, 672) | 0 | block5c_expand_bn[0][0]
block5c_dwconv (DepthwiseConv2D) | (None, 10, 10, 672) | 16,800 | block5c_expand_activation…
block5c_bn (BatchNormalization) | (None, 10, 10, 672) | 2,688 | block5c_dwconv[0][0]
block5c_activation | (None, 10, 10, 672) | 0 | block5c_bn[0][0]

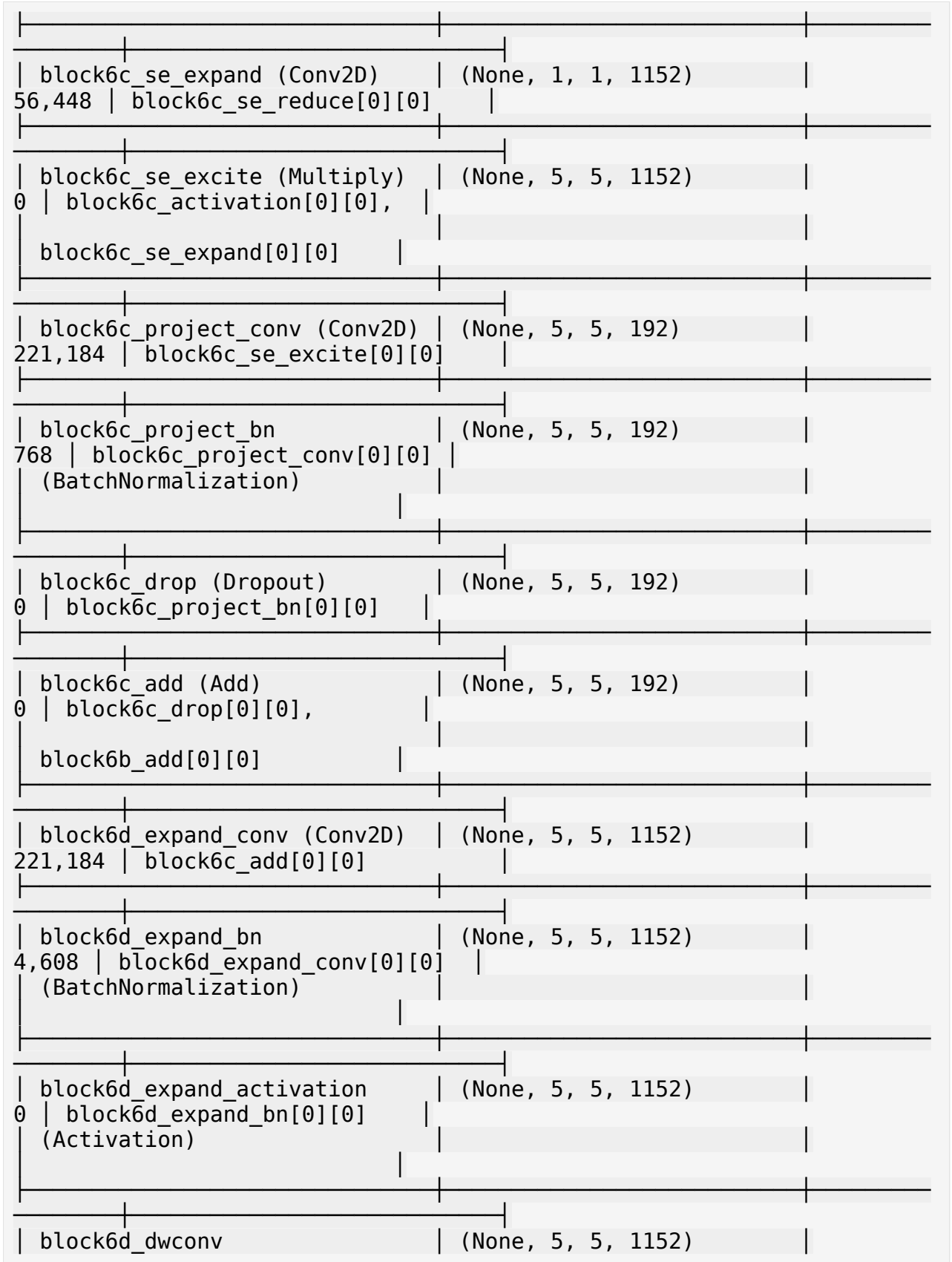


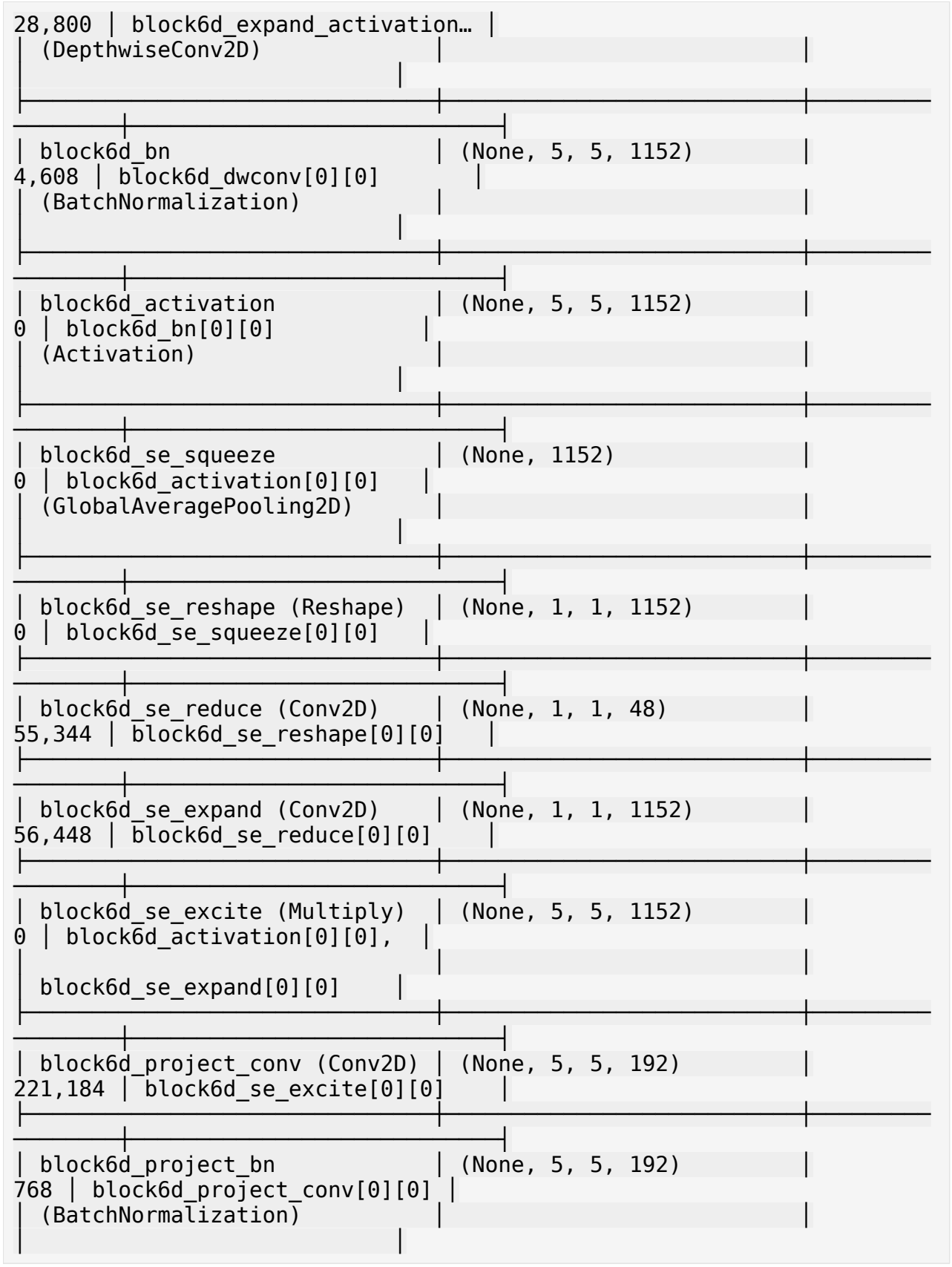

75,264 | block5c_add[0][0]
block6a_expand_bn (BatchNormalization) | (None, 10, 10, 672) | 2,688 | block6a_expand_conv[0][0]
block6a_expand_activation (Activation) | (None, 10, 10, 672) | 0 | block6a_expand_bn[0][0]
block6a_dwconv_pad (ZeroPadding2D) | (None, 13, 13, 672) | 0 | block6a_expand_activation…
block6a_dwconv (DepthwiseConv2D) | (None, 5, 5, 672) | 16,800 | block6a_dwconv_pad[0][0]
block6a_bn (BatchNormalization) | (None, 5, 5, 672) | 2,688 | block6a_dwconv[0][0]
block6a_activation (Activation) | (None, 5, 5, 672) | 0 | block6a_bn[0][0]
block6a_se_squeeze (GlobalAveragePooling2D) | (None, 672) | 0 | block6a_activation[0][0]
block6a_se_reshape (Reshape) | (None, 1, 1, 672) | 0 | block6a_se_squeeze[0][0]

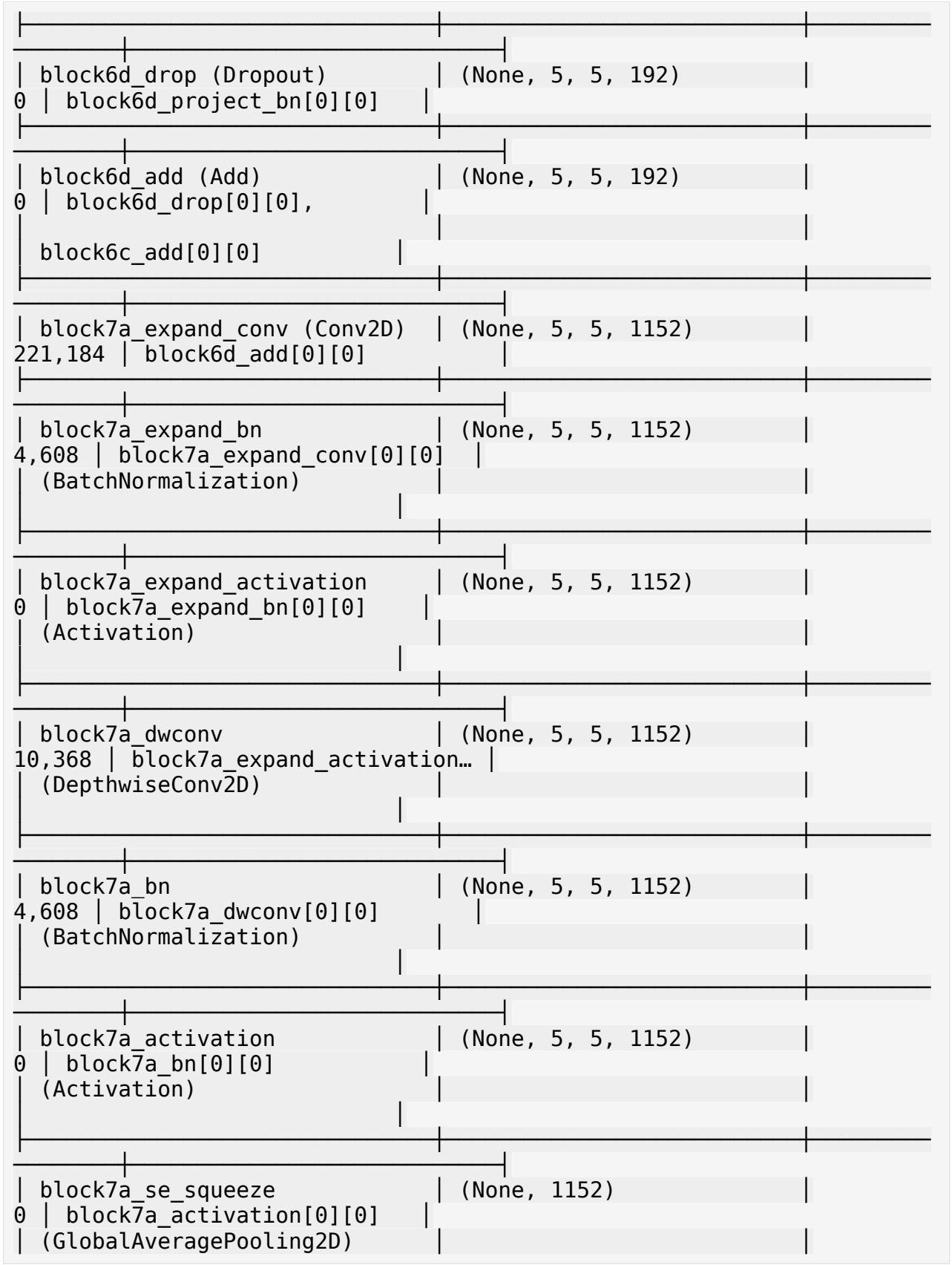

block6a_se_reduce (Conv2D) (None, 1, 1, 28) 18,844 block6a_se_reshape[0][0]
block6a_se_expand (Conv2D) (None, 1, 1, 672) 19,488 block6a_se_reduce[0][0]
block6a_se_excite (Multiply) (None, 5, 5, 672) 0 block6a_activation[0][0], block6a_se_expand[0][0]
block6a_project_conv (Conv2D) (None, 5, 5, 192) 129,024 block6a_se_excite[0][0]
block6a_project_bn (BatchNormalization) (None, 5, 5, 192) 768 block6a_project_conv[0][0]
block6b_expand_conv (Conv2D) (None, 5, 5, 1152) 221,184 block6a_project_bn[0][0]
block6b_expand_bn (BatchNormalization) (None, 5, 5, 1152) 4,608 block6b_expand_conv[0][0]
block6b_expand_activation (Activation) (None, 5, 5, 1152) 0 block6b_expand_bn[0][0]
block6b_dwconv (DepthwiseConv2D) (None, 5, 5, 1152) 28,800 block6b_expand_activation…
block6b_bn (BatchNormalization) (None, 5, 5, 1152) 4,608 block6b_dwconv[0][0]

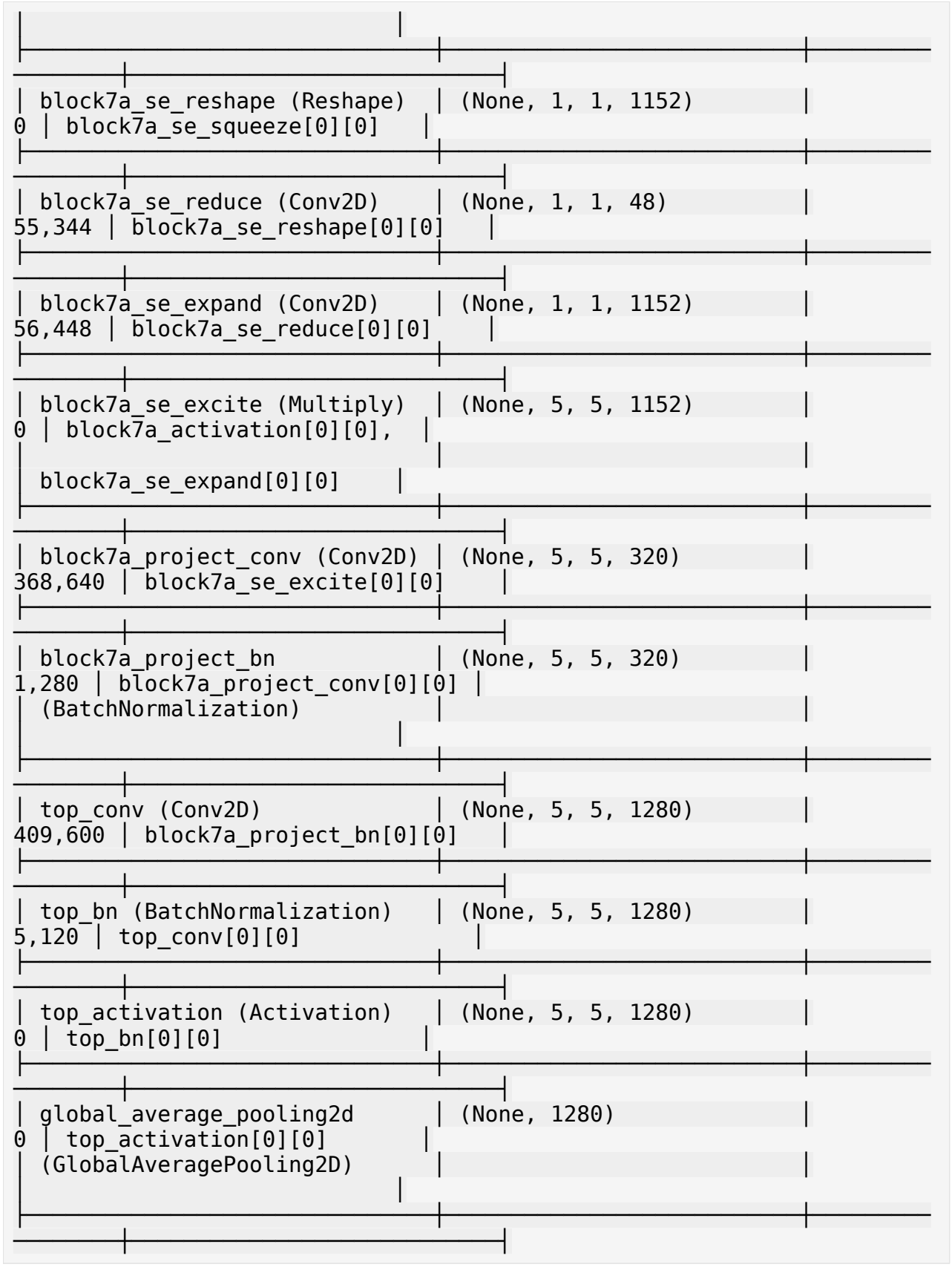

block6b_activation (Activation) (None, 5, 5, 1152) 0 block6b_bn[0][0]
block6b_se_squeeze (GlobalAveragePooling2D) (None, 1152) 0 block6b_activation[0][0]
block6b_se_reshape (Reshape) (None, 1, 1, 1152) 0 block6b_se_squeeze[0][0]
block6b_se_reduce (Conv2D) (None, 1, 1, 48) 55,344 block6b_se_reshape[0][0]
block6b_se_expand (Conv2D) (None, 1, 1, 1152) 56,448 block6b_se_reduce[0][0]
block6b_se_excite (Multiply) (None, 5, 5, 1152) 0 block6b_activation[0][0], block6b_se_expand[0][0]
block6b_project_conv (Conv2D) (None, 5, 5, 192) 221,184 block6b_se_excite[0][0]
block6b_project_bn (BatchNormalization) (None, 5, 5, 192) 768 block6b_project_conv[0][0]
block6b_drop (Dropout) (None, 5, 5, 192) 0 block6b_project_bn[0][0]
block6b_add (Add) (None, 5, 5, 192) 0 block6b_drop[0][0],


block6a_project_bn[0][0]
block6c_expand_conv (Conv2D) (None, 5, 5, 1152) 221,184 block6b_add[0][0]
block6c_expand_bn (BatchNormalization) (None, 5, 5, 1152) 4,608 block6c_expand_conv[0][0]
block6c_expand_activation (Activation) (None, 5, 5, 1152) 0 block6c_expand_bn[0][0]
block6c_dwconv (DepthwiseConv2D) (None, 5, 5, 1152) 28,800 block6c_expand_activation…
block6c_bn (BatchNormalization) (None, 5, 5, 1152) 4,608 block6c_dwconv[0][0]
block6c_activation (Activation) (None, 5, 5, 1152) 0 block6c_bn[0][0]
block6c_se_squeeze (GlobalAveragePooling2D) (None, 1152) 0 block6c_activation[0][0]
block6c_se_reshape (Reshape) (None, 1, 1, 1152) 0 block6c_se_squeeze[0][0]
block6c_se_reduce (Conv2D) (None, 1, 1, 48) 55,344 block6c_se_reshape[0][0]


block6c_se_expand (Conv2D)	(None, 1, 1, 1152)	56,448	block6c_se_reduce[0][0]
block6c_se_excite (Multiply)	(None, 5, 5, 1152)	0	block6c_activation[0][0], block6c_se_expand[0][0]
block6c_project_conv (Conv2D)	(None, 5, 5, 192)	221,184	block6c_se_excite[0][0]
block6c_project_bn (BatchNormalization)	(None, 5, 5, 192)	768	block6c_project_conv[0][0]
block6c_drop (Dropout)	(None, 5, 5, 192)	0	block6c_project_bn[0][0]
block6c_add (Add)	(None, 5, 5, 192)	0	block6c_drop[0][0], block6b_add[0][0]
block6d_expand_conv (Conv2D)	(None, 5, 5, 1152)	221,184	block6c_add[0][0]
block6d_expand_bn (BatchNormalization)	(None, 5, 5, 1152)	4,608	block6d_expand_conv[0][0]
block6d_expand_activation (Activation)	(None, 5, 5, 1152)	0	block6d_expand_bn[0][0]
block6d_dwconv	(None, 5, 5, 1152)




block6d_drop (Dropout) (None, 5, 5, 192) 0 block6d_project_bn[0][0]
block6d_add (Add) (None, 5, 5, 192) 0 block6d_drop[0][0], block6c_add[0][0]
block7a_expand_conv (Conv2D) (None, 5, 5, 1152) 221,184 block6d_add[0][0]
block7a_expand_bn (BatchNormalization) (None, 5, 5, 1152) 4,608 block7a_expand_conv[0][0]
block7a_expand_activation (Activation) (None, 5, 5, 1152) 0 block7a_expand_bn[0][0]
block7a_dwconv (DepthwiseConv2D) (None, 5, 5, 1152) 10,368 block7a_expand_activation…
block7a_bn (BatchNormalization) (None, 5, 5, 1152) 4,608 block7a_dwconv[0][0]
block7a_activation (Activation) (None, 5, 5, 1152) 0 block7a_bn[0][0]
block7a_se_squeeze (GlobalAveragePooling2D) (None, 1152) 0 block7a_activation[0][0]



```
| dropout (Dropout)               | (None, 1280)                 |
0 | global_average_pooling2d[… |
├─────────┼──────────────────────────────┼──────────
──────┼──────────────────────────────┤
| dense (Dense)                   | (None, 5)                    |
6,405 | dropout[0][0]                  |
└─────────┴──────────────────────────────┴──────────
──────┴──────────────────────────────┘

 Total params: 4,055,976 (15.47 MB)

 Trainable params: 4,013,953 (15.31 MB)

 Non-trainable params: 42,023 (164.16 KB)
```

We finally compile our model.

```
model.compile(loss='categorical_crossentropy',optimizer = 'Adam',
metrics= ['accuracy'])

tensorboard = TensorBoard(log_dir = 'logs')
checkpoint = ModelCheckpoint("skin-
snehit.h5",monitor="val_accuracy",save_best_only=True,mode="auto",verb
ose=1)
reduce_lr = ReduceLROnPlateau(monitor = 'val_accuracy', factor = 0.3,
patience = 2, min_delta = 0.001,
                              mode='auto',verbose=1)

history = model.fit(X_train,y_train,validation_split=0.1, epochs =30,
verbose=1, batch_size=16,
                    callbacks=[tensorboard,checkpoint,reduce_lr])

Epoch 1/30
52/52 [==============================] - ETA: 0s - loss: 1.2499 -
accuracy: 0.5342
Epoch 1: val_accuracy improved from -inf to 0.54945, saving model to
skin-snehit.h5
52/52 [==============================] - 18s 150ms/step - loss: 1.2499
- accuracy: 0.5342 - val_loss: 1.2890 - val_accuracy: 0.5495 - lr:
0.0010
Epoch 2/30
51/52 [============================>.] - ETA: 0s - loss: 0.5888 -
accuracy: 0.7904
Epoch 2: val_accuracy improved from 0.54945 to 0.63736, saving model
to skin-snehit.h5
52/52 [==============================] - 5s 100ms/step - loss: 0.5929
- accuracy: 0.7885 - val_loss: 1.2271 - val_accuracy: 0.6374 - lr:
0.0010
Epoch 3/30
51/52 [============================>.] - ETA: 0s - loss: 0.4432 -
```

```
accuracy: 0.8493
Epoch 3: val_accuracy did not improve from 0.63736
52/52 [==============================] - 4s 86ms/step - loss: 0.4432 -
accuracy: 0.8496 - val_loss: 1.3612 - val_accuracy: 0.6264 - lr:
0.0010
Epoch 4/30
51/52 [============================>.] - ETA: 0s - loss: 0.2475 -
accuracy: 0.9118
Epoch 4: val_accuracy improved from 0.63736 to 0.79121, saving model
to skin-snehit.h5
52/52 [==============================] - 5s 99ms/step - loss: 0.2487 -
accuracy: 0.9108 - val_loss: 0.8117 - val_accuracy: 0.7912 - lr:
0.0010
Epoch 5/30
51/52 [============================>.] - ETA: 0s - loss: 0.2187 -
accuracy: 0.9203
Epoch 5: val_accuracy did not improve from 0.79121
52/52 [==============================] - 4s 85ms/step - loss: 0.2219 -
accuracy: 0.9193 - val_loss: 1.0181 - val_accuracy: 0.7582 - lr:
0.0010
Epoch 6/30
51/52 [============================>.] - ETA: 0s - loss: 0.2151 -
accuracy: 0.9191
Epoch 6: val_accuracy did not improve from 0.79121

Epoch 6: ReduceLROnPlateau reducing learning rate to
0.0003000000142492354.
52/52 [==============================] - 4s 86ms/step - loss: 0.2166 -
accuracy: 0.9193 - val_loss: 0.8968 - val_accuracy: 0.7582 - lr:
0.0010
Epoch 7/30
51/52 [============================>.] - ETA: 0s - loss: 0.1594 -
accuracy: 0.9485
Epoch 7: val_accuracy improved from 0.79121 to 0.84615, saving model
to skin-snehit.h5
52/52 [==============================] - 5s 100ms/step - loss: 0.1599
- accuracy: 0.9487 - val_loss: 0.6371 - val_accuracy: 0.8462 - lr:
3.0000e-04
Epoch 8/30
51/52 [============================>.] - ETA: 0s - loss: 0.0816 -
accuracy: 0.9755
Epoch 8: val_accuracy did not improve from 0.84615
52/52 [==============================] - 4s 86ms/step - loss: 0.0835 -
accuracy: 0.9743 - val_loss: 0.4981 - val_accuracy: 0.8462 - lr:
3.0000e-04
Epoch 9/30
51/52 [============================>.] - ETA: 0s - loss: 0.0623 -
accuracy: 0.9804
Epoch 9: val_accuracy improved from 0.84615 to 0.85714, saving model
```

```
to skin-snehit.h5
52/52 [==============================] - 5s 99ms/step - loss: 0.0636 -
accuracy: 0.9804 - val_loss: 0.5931 - val_accuracy: 0.8571 - lr:
3.0000e-04
Epoch 10/30
51/52 [============================>.] - ETA: 0s - loss: 0.0575 -
accuracy: 0.9841
Epoch 10: val_accuracy improved from 0.85714 to 0.89011, saving model
to skin-snehit.h5
52/52 [==============================] - 5s 100ms/step - loss: 0.0593
- accuracy: 0.9829 - val_loss: 0.5514 - val_accuracy: 0.8901 - lr:
3.0000e-04
Epoch 11/30
51/52 [============================>.] - ETA: 0s - loss: 0.0474 -
accuracy: 0.9865
Epoch 11: val_accuracy did not improve from 0.89011
52/52 [==============================] - 4s 86ms/step - loss: 0.0535 -
accuracy: 0.9853 - val_loss: 0.5679 - val_accuracy: 0.8462 - lr:
3.0000e-04
Epoch 12/30
51/52 [============================>.] - ETA: 0s - loss: 0.0239 -
accuracy: 0.9939
Epoch 12: val_accuracy did not improve from 0.89011

Epoch 12: ReduceLROnPlateau reducing learning rate to
9.000000427477062e-05.
52/52 [==============================] - 5s 87ms/step - loss: 0.0242 -
accuracy: 0.9939 - val_loss: 0.6206 - val_accuracy: 0.8462 - lr:
3.0000e-04
Epoch 13/30
51/52 [============================>.] - ETA: 0s - loss: 0.0229 -
accuracy: 0.9975
Epoch 13: val_accuracy did not improve from 0.89011
52/52 [==============================] - 5s 87ms/step - loss: 0.0234 -
accuracy: 0.9976 - val_loss: 0.6314 - val_accuracy: 0.8352 - lr:
9.0000e-05
Epoch 14/30
51/52 [============================>.] - ETA: 0s - loss: 0.0188 -
accuracy: 0.9988
Epoch 14: val_accuracy did not improve from 0.89011

Epoch 14: ReduceLROnPlateau reducing learning rate to
2.700000040931627e-05.
52/52 [==============================] - 4s 86ms/step - loss: 0.0225 -
accuracy: 0.9963 - val_loss: 0.6357 - val_accuracy: 0.8352 - lr:
9.0000e-05
Epoch 15/30
51/52 [============================>.] - ETA: 0s - loss: 0.0274 -
accuracy: 0.9939
```

```
Epoch 15: val_accuracy did not improve from 0.89011
52/52 [==============================] - 5s 88ms/step - loss: 0.0276 -
accuracy: 0.9939 - val_loss: 0.6342 - val_accuracy: 0.8352 - lr:
2.7000e-05
Epoch 16/30
51/52 [============================>.] - ETA: 0s - loss: 0.0147 -
accuracy: 0.9975
Epoch 16: val_accuracy did not improve from 0.89011

Epoch 16: ReduceLROnPlateau reducing learning rate to
8.100000013655517e-06.
52/52 [==============================] - 5s 88ms/step - loss: 0.0154 -
accuracy: 0.9976 - val_loss: 0.6291 - val_accuracy: 0.8352 - lr:
2.7000e-05
Epoch 17/30
52/52 [==============================] - ETA: 0s - loss: 0.0260 -
accuracy: 0.9951
Epoch 17: val_accuracy did not improve from 0.89011
52/52 [==============================] - 5s 94ms/step - loss: 0.0260 -
accuracy: 0.9951 - val_loss: 0.6276 - val_accuracy: 0.8352 - lr:
8.1000e-06
Epoch 18/30
51/52 [============================>.] - ETA: 0s - loss: 0.0240 -
accuracy: 0.9939
Epoch 18: val_accuracy did not improve from 0.89011

Epoch 18: ReduceLROnPlateau reducing learning rate to
2.429999949526973e-06.
52/52 [==============================] - 6s 108ms/step - loss: 0.0295
- accuracy: 0.9914 - val_loss: 0.6305 - val_accuracy: 0.8352 - lr:
8.1000e-06
Epoch 19/30
51/52 [============================>.] - ETA: 0s - loss: 0.0164 -
accuracy: 0.9963
Epoch 19: val_accuracy did not improve from 0.89011
52/52 [==============================] - 5s 88ms/step - loss: 0.0176 -
accuracy: 0.9963 - val_loss: 0.6335 - val_accuracy: 0.8352 - lr:
2.4300e-06
Epoch 20/30
51/52 [============================>.] - ETA: 0s - loss: 0.0186 -
accuracy: 0.9975
Epoch 20: val_accuracy did not improve from 0.89011

Epoch 20: ReduceLROnPlateau reducing learning rate to
7.289999985005124e-07.
52/52 [==============================] - 5s 87ms/step - loss: 0.0249 -
accuracy: 0.9951 - val_loss: 0.6268 - val_accuracy: 0.8352 - lr:
2.4300e-06
Epoch 21/30
51/52 [============================>.] - ETA: 0s - loss: 0.0204 -
```

```
accuracy: 0.9975
Epoch 21: val_accuracy did not improve from 0.89011
52/52 [==============================] - 5s 88ms/step - loss: 0.0209 -
accuracy: 0.9976 - val_loss: 0.6309 - val_accuracy: 0.8352 - lr:
7.2900e-07
Epoch 22/30
51/52 [============================>.] - ETA: 0s - loss: 0.0192 -
accuracy: 0.9939
Epoch 22: val_accuracy did not improve from 0.89011

Epoch 22: ReduceLROnPlateau reducing learning rate to
2.1870000637136398e-07.
52/52 [==============================] - 5s 89ms/step - loss: 0.0199 -
accuracy: 0.9939 - val_loss: 0.6279 - val_accuracy: 0.8352 - lr:
7.2900e-07
Epoch 23/30
51/52 [============================>.] - ETA: 0s - loss: 0.0174 -
accuracy: 0.9963
Epoch 23: val_accuracy did not improve from 0.89011
52/52 [==============================] - 5s 102ms/step - loss: 0.0235
- accuracy: 0.9939 - val_loss: 0.6261 - val_accuracy: 0.8352 - lr:
2.1870e-07
Epoch 24/30
51/52 [============================>.] - ETA: 0s - loss: 0.0197 -
accuracy: 0.9963
Epoch 24: val_accuracy did not improve from 0.89011

Epoch 24: ReduceLROnPlateau reducing learning rate to
6.561000276406048e-08.
52/52 [==============================] - 5s 87ms/step - loss: 0.0212 -
accuracy: 0.9963 - val_loss: 0.6256 - val_accuracy: 0.8352 - lr:
2.1870e-07
Epoch 25/30
51/52 [============================>.] - ETA: 0s - loss: 0.0220 -
accuracy: 0.9951
Epoch 25: val_accuracy did not improve from 0.89011
52/52 [==============================] - 5s 88ms/step - loss: 0.0240 -
accuracy: 0.9939 - val_loss: 0.6197 - val_accuracy: 0.8352 - lr:
6.5610e-08
Epoch 26/30
52/52 [==============================] - ETA: 0s - loss: 0.0172 -
accuracy: 0.9963
Epoch 26: val_accuracy did not improve from 0.89011

Epoch 26: ReduceLROnPlateau reducing learning rate to
1.9683000829218145e-08.
52/52 [==============================] - 5s 88ms/step - loss: 0.0172 -
accuracy: 0.9963 - val_loss: 0.6262 - val_accuracy: 0.8352 - lr:
6.5610e-08
Epoch 27/30
```

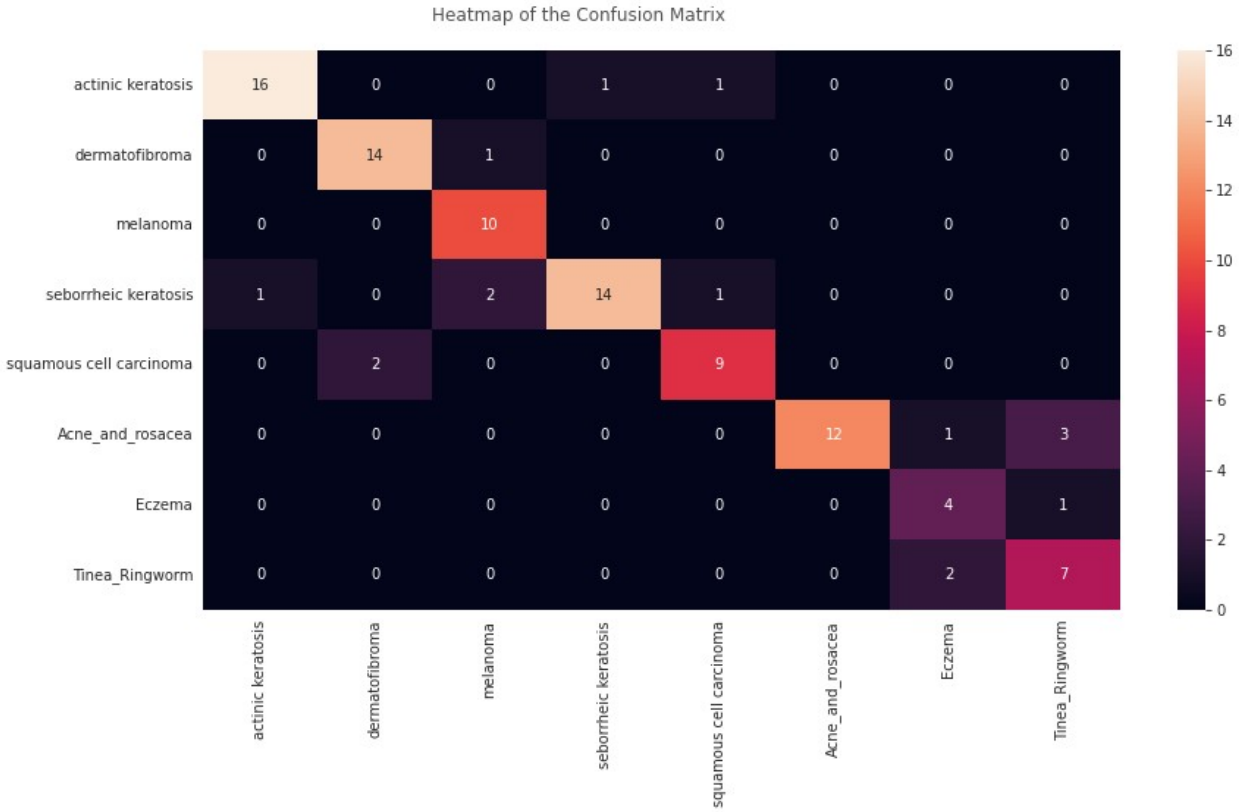
```
51/52 [============================>.] - ETA: 0s - loss: 0.0191 -
accuracy: 0.9988
Epoch 27: val_accuracy did not improve from 0.89011
52/52 [==============================] - 5s 88ms/step - loss: 0.0215 -
accuracy: 0.9976 - val_loss: 0.6277 - val_accuracy: 0.8352 - lr:
1.9683e-08
Epoch 28/30
51/52 [============================>.] - ETA: 0s - loss: 0.0204 -
accuracy: 0.9963
Epoch 28: val_accuracy did not improve from 0.89011

Epoch 28: ReduceLROnPlateau reducing learning rate to
5.904900035602622e-09.
52/52 [==============================] - 5s 87ms/step - loss: 0.0229 -
accuracy: 0.9951 - val_loss: 0.6313 - val_accuracy: 0.8352 - lr:
1.9683e-08
Epoch 29/30
51/52 [============================>.] - ETA: 0s - loss: 0.0214 -
accuracy: 0.9939
Epoch 29: val_accuracy did not improve from 0.89011
52/52 [==============================] - 5s 88ms/step - loss: 0.0241 -
accuracy: 0.9927 - val_loss: 0.6275 - val_accuracy: 0.8352 - lr:
5.9049e-09
Epoch 30/30
51/52 [============================>.] - ETA: 0s - loss: 0.0159 -
accuracy: 0.9988
Epoch 30: val_accuracy did not improve from 0.89011

Epoch 30: ReduceLROnPlateau reducing learning rate to
1.7714700373261393e-09.
52/52 [==============================] - 5s 88ms/step - loss: 0.0161 -
accuracy: 0.9988 - val_loss: 0.6271 - val_accuracy: 0.8352 - lr:
5.9049e-09
```

```
pred = model.predict(X_test)
pred = np.argmax(pred,axis=1)
y_test_new = np.argmax(y_test,axis=1)
print(classification_report(y_test_new,pred))
```

```
4/4 [==============================] - 3s 214ms/step
              precision    recall  f1-score   support

           0       0.94      0.89      0.91        18
           1       0.88      0.93      0.90        15
           2       0.77      1.00      0.87        10
           3       0.93      0.78      0.85        18
           4       0.82      0.82      0.82        11
           5       1.00      0.75      0.86        16
           6       0.57      0.80      0.67         5
           7       0.64      0.78      0.70         9
```

```
    accuracy                           0.84       102
   macro avg       0.82      0.84      0.82       102
weighted avg       0.86      0.84      0.85       102
```

```
fig,ax=plt.subplots(1,1,figsize=(14,7))
sns.heatmap(confusion_matrix(y_test_new,pred),ax=ax,xticklabels=labels
,yticklabels=labels,annot=True)
fig.text(s='Heatmap of the Confusion
Matrix',size=12,y=0.92,x=0.28,alpha=0.8)

plt.show()
```




```
model.save('snehit-skin.keras')
```

Table for Training Dataset

```
pred = model.predict(X_train)
pred = np.argmax(pred,axis=1)
y_train_new = np.argmax(y_train,axis=1)

matrix = confusion_matrix(y_train_new,pred)
matrix.diagonal()/matrix.sum(axis=1)
```

```
array([0.98319328, 0.96551724, 0.99145299, 1.        , 1.        ])
```

```
import math
print(91, 89, 91, 76, 87)
print(math.ceil(0.98319328*91), math.ceil(0.96551724*89),
math.ceil(0.99145299*91), math.ceil(1.0*76), math.ceil(1.0*87))
print(math.ceil(0.98319328*100), math.ceil(0.96551724*100),
math.ceil(0.99145299*100), math.ceil(1.0*100), math.ceil(1.0*100))
```

```
91 89 91 76 87
90 86 91 76 87
99 97 100 100 100
```

Table for Testing Dataset

```
pred = model.predict(X_test)
pred = np.argmax(pred,axis=1)
y_test_new = np.argmax(y_test,axis=1)

matrix = confusion_matrix(y_test_new,pred)
matrix.diagonal()/matrix.sum(axis=1)
```

```
array([1.        , 0.91666667, 0.69230769, 1.        , 0.73333333])
```

```
import math
print(39, 39, 39, 40, 40)
print(math.ceil(1.0*39), math.ceil(0.91666667*39),
math.ceil(0.69230769*39), math.ceil(1.0*40), math.ceil(0.73333333*40))
print(math.ceil(1.0*100), math.ceil(0.91666667*100),
math.ceil(0.69230769*100), math.ceil(1.0*100),
math.ceil(0.73333333*100))
```

```
39 39 39 40 40
39 36 27 40 30
100 92 70 100 74
```

Transfer Learning _ Inception V3

```
import numpy as np
from keras_preprocessing.image import ImageDataGenerator
import tensorflow as tf
import keras
import pandas as pd
from keras.applications.inception_resnet_v2 import InceptionResNetV2
from keras.models import Model
from keras.applications.inception_v3 import preprocess_input
from keras.applications.inception_v3 import InceptionV3
```

```
from keras.layers import Dense, BatchNormalization,
GlobalAveragePooling2D, Conv2D, Flatten, MaxPooling2D, Dropout
from tensorflow.keras.optimizers import Adam, Nadam

from keras.applications.inception_v3 import InceptionV3

tf.keras.backend.clear_session()

# We use model inceptionV3
pre_train_model = InceptionV3(
      include_top = False,
      weights = "imagenet",
      input_shape = (image_size,image_size, 3)
)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/inception_v3/
inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5
87910968/87910968 [==============================] - 0s 0us/step
```

```
# pre_train_model.summary()

# just use a part of model because our task is simple so if use whole
model will be overfitting

for layer in pre_train_model.layers:
    layer.trainable = False
last_layer = pre_train_model.get_layer('mixed9')  # cut begin to layer
block8_9_mixed
last_output = pre_train_model.output

# Add some custom layer to do our task, output will be 1 node
# x = MaxPooling2D(pool_size=(2,2))(last_output)
x = Flatten()(last_output)
# x = Dense(2048, activation='relu')(x)
# x = Dense(1024, activation='relu')(x)
# x = Dense(1024, activation='relu')(x)
x = Dense(512, activation='relu')(x)
# x = Dense(512, activation='relu')(x)
# x = Dense(256, activation='relu')(x)
x = Dense(256, activation='relu')(x)
x = Dense(128, activation='relu')(x)
x = Dense(64, activation='relu')(x)
x = Dropout(0.3)(x)
output = Dense(5, activation='softmax')(x)

# Define optimizer, learning rate and loss function
model = Model(pre_train_model.input, output)
model.compile(optimizer=Adam(learning_rate=0.0001),loss='categorical_c
rossentropy',metrics=['acc'])
```

```
tensorboard = TensorBoard(log_dir = 'logs')
checkpoint = ModelCheckpoint("Inception V3-model-
28.h5",monitor="val_accuracy",save_best_only=True,mode="auto",verbose=
1)
reduce_lr = ReduceLROnPlateau(monitor = 'val_accuracy', factor = 0.3,
patience = 2, min_delta = 0.001,
                              mode='auto',verbose=1)

history = model.fit(X_train,y_train,validation_split=0.1, epochs =28,
verbose=1, batch_size=16,
                    callbacks=[tensorboard,checkpoint,reduce_lr])

Epoch 1/28
52/52 [==============================] - ETA: 0s - loss: 13.9524 -
acc: 0.2005

WARNING:tensorflow:Can save best model only with val_accuracy
available, skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are:
loss,acc,val_loss,val_acc,lr

 52/52 [==============================] - 9s 78ms/step - loss: 13.9524
- acc: 0.2005 - val_loss: 5.1864 - val_acc: 0.2418 - lr: 1.0000e-04
Epoch 2/28
51/52 [============================>.] - ETA: 0s - loss: 5.1725 - acc:
0.2610

WARNING:tensorflow:Can save best model only with val_accuracy
available, skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are:
loss,acc,val_loss,val_acc,lr

 52/52 [==============================] - 1s 27ms/step - loss: 5.1670
- acc: 0.2616 - val_loss: 2.6734 - val_acc: 0.3516 - lr: 1.0000e-04
Epoch 3/28
52/52 [==============================] - ETA: 0s - loss: 2.5343 - acc:
0.2653

WARNING:tensorflow:Can save best model only with val_accuracy
available, skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are:
loss,acc,val_loss,val_acc,lr

 52/52 [==============================] - 1s 28ms/step - loss: 2.5343
- acc: 0.2653 - val_loss: 2.1043 - val_acc: 0.3297 - lr: 1.0000e-04
Epoch 4/28
```

```
51/52 [============================>.] - ETA: 0s - loss: 1.9144 - acc:
0.2953

WARNING:tensorflow:Can save best model only with val_accuracy
available, skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are:
loss,acc,val_loss,val_acc,lr

 52/52 [==============================] - 1s 27ms/step - loss: 1.9140
- acc: 0.2946 - val_loss: 1.6066 - val_acc: 0.3297 - lr: 1.0000e-04
Epoch 5/28
51/52 [============================>.] - ETA: 0s - loss: 1.7732 - acc:
0.2978

WARNING:tensorflow:Can save best model only with val_accuracy
available, skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are:
loss,acc,val_loss,val_acc,lr

 52/52 [==============================] - 2s 36ms/step - loss: 1.7696
- acc: 0.2995 - val_loss: 1.5903 - val_acc: 0.3956 - lr: 1.0000e-04
Epoch 6/28
51/52 [============================>.] - ETA: 0s - loss: 1.6796 - acc:
0.3309

WARNING:tensorflow:Can save best model only with val_accuracy
available, skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are:
loss,acc,val_loss,val_acc,lr

 52/52 [==============================] - 1s 27ms/step - loss: 1.6802
- acc: 0.3301 - val_loss: 1.7313 - val_acc: 0.3516 - lr: 1.0000e-04
Epoch 7/28
51/52 [============================>.] - ETA: 0s - loss: 1.6369 - acc:
0.3493

WARNING:tensorflow:Can save best model only with val_accuracy
available, skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are:
loss,acc,val_loss,val_acc,lr

 52/52 [==============================] - 1s 27ms/step - loss: 1.6424
- acc: 0.3484 - val_loss: 1.8127 - val_acc: 0.3297 - lr: 1.0000e-04
Epoch 8/28
52/52 [==============================] - ETA: 0s - loss: 1.6408 - acc:
0.3533
```

```
WARNING:tensorflow:Can save best model only with val_accuracy
available, skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are:
loss,acc,val_loss,val_acc,lr

 52/52 [==============================] - 1s 27ms/step - loss: 1.6408
- acc: 0.3533 - val_loss: 1.9458 - val_acc: 0.3516 - lr: 1.0000e-04
Epoch 9/28
50/52 [============================>..] - ETA: 0s - loss: 1.5143 - acc:
0.4050

WARNING:tensorflow:Can save best model only with val_accuracy
available, skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are:
loss,acc,val_loss,val_acc,lr

 52/52 [==============================] - 1s 28ms/step - loss: 1.5163
- acc: 0.4046 - val_loss: 1.9769 - val_acc: 0.4286 - lr: 1.0000e-04
Epoch 10/28
51/52 [============================>.] - ETA: 0s - loss: 1.6027 - acc:
0.3701

WARNING:tensorflow:Can save best model only with val_accuracy
available, skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are:
loss,acc,val_loss,val_acc,lr

 52/52 [==============================] - 1s 27ms/step - loss: 1.6016
- acc: 0.3716 - val_loss: 2.0208 - val_acc: 0.4286 - lr: 1.0000e-04
Epoch 11/28
50/52 [============================>..] - ETA: 0s - loss: 1.5641 - acc:
0.4100

WARNING:tensorflow:Can save best model only with val_accuracy
available, skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are:
loss,acc,val_loss,val_acc,lr

 52/52 [==============================] - 1s 27ms/step - loss: 1.5595
- acc: 0.4095 - val_loss: 1.7443 - val_acc: 0.3956 - lr: 1.0000e-04
Epoch 12/28
51/52 [============================>.] - ETA: 0s - loss: 1.5762 - acc:
0.3787

WARNING:tensorflow:Can save best model only with val_accuracy
available, skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
```

```
`val_accuracy` which is not available. Available metrics are:
loss,acc,val_loss,val_acc,lr

 52/52 [==============================] - 1s 27ms/step - loss: 1.5731
- acc: 0.3802 - val_loss: 2.2673 - val_acc: 0.2857 - lr: 1.0000e-04
Epoch 13/28
50/52 [===========================>..] - ETA: 0s - loss: 1.6884 - acc:
0.4013

WARNING:tensorflow:Can save best model only with val_accuracy
available, skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are:
loss,acc,val_loss,val_acc,lr

 52/52 [==============================] - 1s 28ms/step - loss: 1.6838
- acc: 0.4010 - val_loss: 1.6791 - val_acc: 0.4615 - lr: 1.0000e-04
Epoch 14/28
51/52 [============================>.] - ETA: 0s - loss: 1.5473 - acc:
0.3934

WARNING:tensorflow:Can save best model only with val_accuracy
available, skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are:
loss,acc,val_loss,val_acc,lr

 52/52 [==============================] - 1s 27ms/step - loss: 1.5459
- acc: 0.3936 - val_loss: 1.7177 - val_acc: 0.3297 - lr: 1.0000e-04
Epoch 15/28
51/52 [============================>.] - ETA: 0s - loss: 1.6255 - acc:
0.3738

WARNING:tensorflow:Can save best model only with val_accuracy
available, skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are:
loss,acc,val_loss,val_acc,lr

 52/52 [==============================] - 1s 27ms/step - loss: 1.6285
- acc: 0.3729 - val_loss: 1.8195 - val_acc: 0.3077 - lr: 1.0000e-04
Epoch 16/28
50/52 [===========================>..] - ETA: 0s - loss: 1.6030 - acc:
0.3862

WARNING:tensorflow:Can save best model only with val_accuracy
available, skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are:
loss,acc,val_loss,val_acc,lr
```

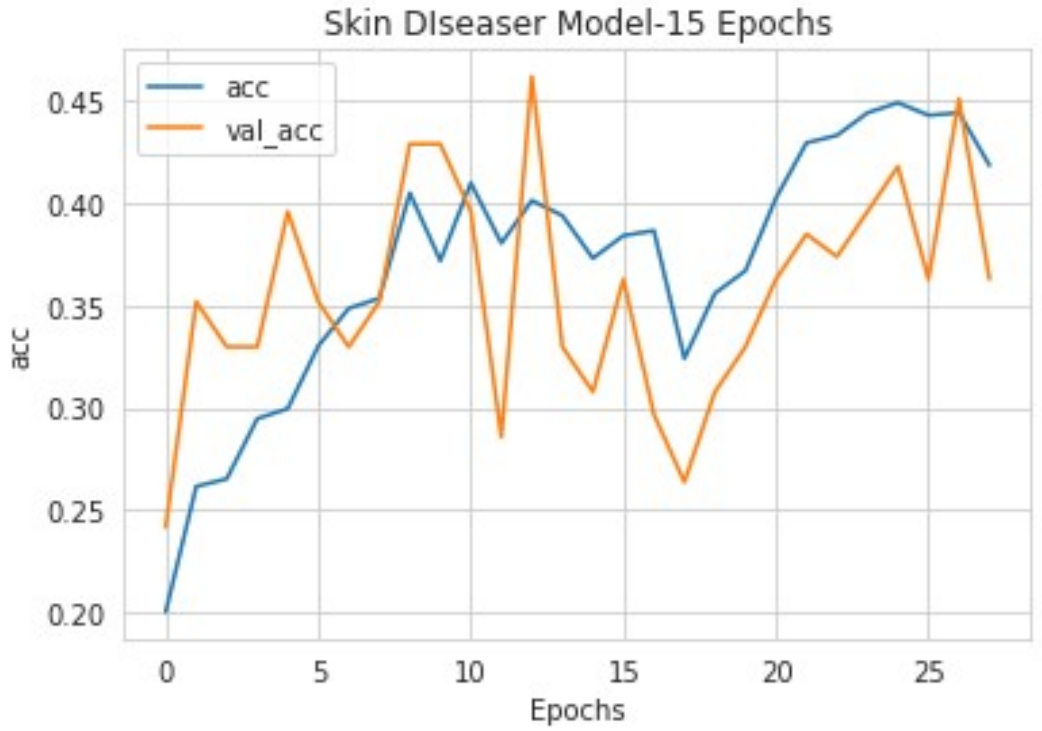
```
 52/52 [==============================] - 1s 28ms/step - loss: 1.6054
- acc: 0.3839 - val_loss: 2.1730 - val_acc: 0.3626 - lr: 1.0000e-04
Epoch 17/28
50/52 [============================>..] - ETA: 0s - loss: 1.5294 - acc:
0.3900

WARNING:tensorflow:Can save best model only with val_accuracy
available, skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are:
loss,acc,val_loss,val_acc,lr

 52/52 [==============================] - 1s 28ms/step - loss: 1.5341
- acc: 0.3863 - val_loss: 2.0961 - val_acc: 0.2967 - lr: 1.0000e-04
Epoch 18/28
50/52 [============================>..] - ETA: 0s - loss: 1.6593 - acc:
0.3262

WARNING:tensorflow:Can save best model only with val_accuracy
available, skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are:
loss,acc,val_loss,val_acc,lr

 52/52 [==============================] - 1s 28ms/step - loss: 1.6603
- acc: 0.3240 - val_loss: 2.2845 - val_acc: 0.2637 - lr: 1.0000e-04
Epoch 19/28
50/52 [============================>..] - ETA: 0s - loss: 1.5804 - acc:
0.3562

WARNING:tensorflow:Can save best model only with val_accuracy
available, skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are:
loss,acc,val_loss,val_acc,lr

 52/52 [==============================] - 1s 27ms/step - loss: 1.5873
- acc: 0.3557 - val_loss: 1.8465 - val_acc: 0.3077 - lr: 1.0000e-04
Epoch 20/28
51/52 [============================>.] - ETA: 0s - loss: 1.5723 - acc:
0.3676

WARNING:tensorflow:Can save best model only with val_accuracy
available, skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are:
loss,acc,val_loss,val_acc,lr

 52/52 [==============================] - 1s 28ms/step - loss: 1.5743
- acc: 0.3667 - val_loss: 2.4162 - val_acc: 0.3297 - lr: 1.0000e-04
Epoch 21/28
```

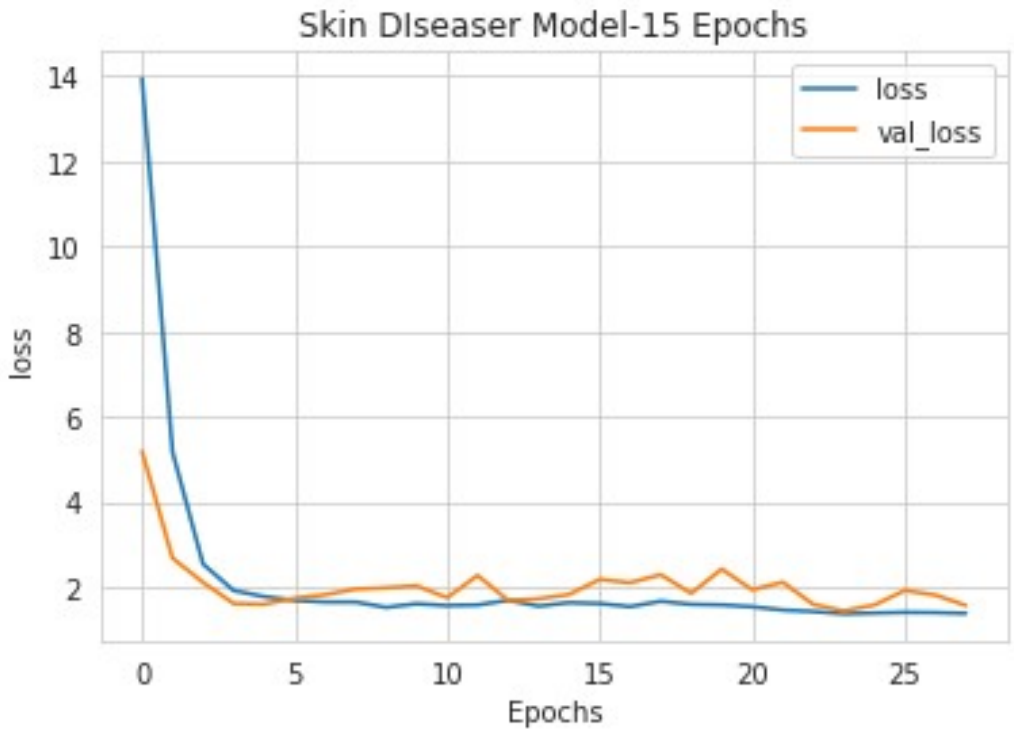
```
50/52 [===========================>..] - ETA: 0s - loss: 1.5197 - acc:
0.4075

WARNING:tensorflow:Can save best model only with val_accuracy
available, skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are:
loss,acc,val_loss,val_acc,lr

 52/52 [==============================] - 1s 27ms/step - loss: 1.5313
- acc: 0.4022 - val_loss: 1.9290 - val_acc: 0.3626 - lr: 1.0000e-04
Epoch 22/28
52/52 [==============================] - ETA: 0s - loss: 1.4540 - acc:
0.4291

WARNING:tensorflow:Can save best model only with val_accuracy
available, skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are:
loss,acc,val_loss,val_acc,lr

 52/52 [==============================] - 1s 27ms/step - loss: 1.4540
- acc: 0.4291 - val_loss: 2.1081 - val_acc: 0.3846 - lr: 1.0000e-04
Epoch 23/28
50/52 [===========================>..] - ETA: 0s - loss: 1.4162 - acc:
0.4350

WARNING:tensorflow:Can save best model only with val_accuracy
available, skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are:
loss,acc,val_loss,val_acc,lr

 52/52 [==============================] - 1s 27ms/step - loss: 1.4191
- acc: 0.4328 - val_loss: 1.5844 - val_acc: 0.3736 - lr: 1.0000e-04
Epoch 24/28
51/52 [============================>.] - ETA: 0s - loss: 1.3646 - acc:
0.4449

WARNING:tensorflow:Can save best model only with val_accuracy
available, skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are:
loss,acc,val_loss,val_acc,lr

 52/52 [==============================] - 1s 28ms/step - loss: 1.3666
- acc: 0.4438 - val_loss: 1.4334 - val_acc: 0.3956 - lr: 1.0000e-04
Epoch 25/28
50/52 [===========================>..] - ETA: 0s - loss: 1.3863 - acc:
0.4450
```

```
WARNING:tensorflow:Can save best model only with val_accuracy
available, skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are:
loss,acc,val_loss,val_acc,lr

 52/52 [==============================] - 1s 28ms/step - loss: 1.3827
- acc: 0.4487 - val_loss: 1.5721 - val_acc: 0.4176 - lr: 1.0000e-04
Epoch 26/28
51/52 [============================>.] - ETA: 0s - loss: 1.4019 - acc:
0.4412

WARNING:tensorflow:Can save best model only with val_accuracy
available, skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are:
loss,acc,val_loss,val_acc,lr

 52/52 [==============================] - 1s 27ms/step - loss: 1.3994
- acc: 0.4425 - val_loss: 1.9141 - val_acc: 0.3626 - lr: 1.0000e-04
Epoch 27/28
50/52 [============================>..] - ETA: 0s - loss: 1.3929 - acc:
0.4425

WARNING:tensorflow:Can save best model only with val_accuracy
available, skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are:
loss,acc,val_loss,val_acc,lr

 52/52 [==============================] - 1s 28ms/step - loss: 1.3936
- acc: 0.4438 - val_loss: 1.8064 - val_acc: 0.4505 - lr: 1.0000e-04
Epoch 28/28
51/52 [============================>.] - ETA: 0s - loss: 1.3761 - acc:
0.4179

WARNING:tensorflow:Can save best model only with val_accuracy
available, skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are:
loss,acc,val_loss,val_acc,lr

 52/52 [==============================] - 1s 27ms/step - loss: 1.3747
- acc: 0.4181 - val_loss: 1.5609 - val_acc: 0.3626 - lr: 1.0000e-04
```

```python
#Visualize Training
def plot_graphs(history, string):
    sns.set_style("whitegrid")
    plt.plot(history.history[string])
    plt.plot(history.history["val_"+string])
    plt.xlabel("Epochs")
```

```
    plt.ylabel(string)
    plt.title("Skin DIseaser Model-15 Epochs")
    plt.legend([string,"val_"+string])
    plt.show()
plot_graphs(history,'acc')
plot_graphs(history,'loss')
```



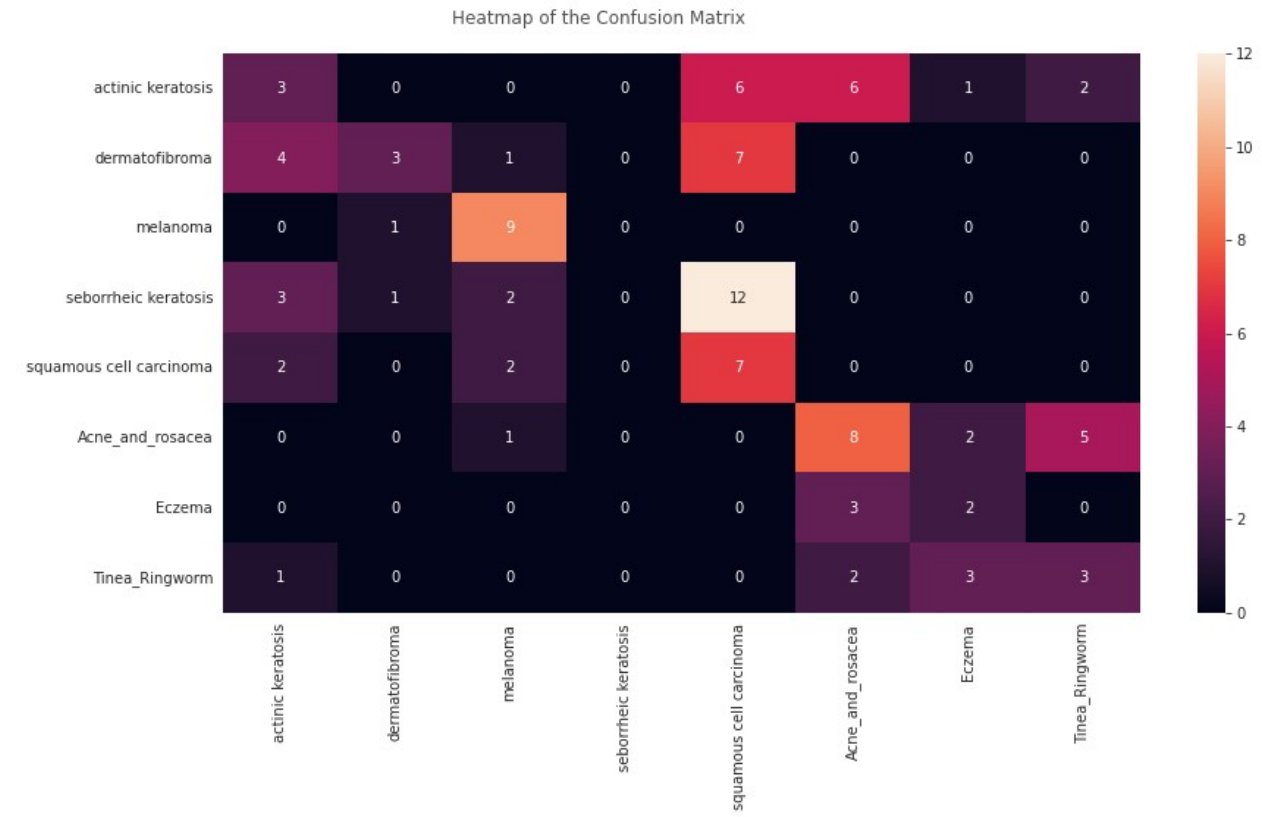



```
pred = model.predict(X_test)
pred = np.argmax(pred,axis=1)
y_test_new = np.argmax(y_test,axis=1)
print(classification_report(y_test_new,pred))
```

```
4/4 [==============================] - 0s 52ms/step
              precision    recall  f1-score   support

           0       0.23      0.17      0.19        18
           1       0.60      0.20      0.30        15
           2       0.60      0.90      0.72        10
           3       0.00      0.00      0.00        18
           4       0.22      0.64      0.33        11
           5       0.42      0.50      0.46        16
           6       0.25      0.40      0.31         5
           7       0.30      0.33      0.32         9

    accuracy                           0.34       102
   macro avg       0.33      0.39      0.33       102
weighted avg       0.32      0.34      0.30       102


/usr/local/lib/python3.8/dist-packages/sklearn/metrics/
_classification.py:1318: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/metrics/_classification
.py:1318: UndefinedMetricWarning: Precision and F-score are ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.8/dist-packages/sklearn/metrics/_classification
.py:1318: UndefinedMetricWarning: Precision and F-score are ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
fig,ax=plt.subplots(1,1,figsize=(14,7))
sns.heatmap(confusion_matrix(y_test_new,pred),ax=ax,xticklabels=labels
,yticklabels=labels,annot=True)
fig.text(s='Heatmap of the Confusion
Matrix',size=12,y=0.92,x=0.28,alpha=0.8)

plt.show()
```



Bonus Content: Widgets

I've made these Widgets in which we can upload images from our local machine and predict whether the MRI scan has a Brain Tumour or not and to classify which Tumor it is. Unfortunately,

it doesn't work on Kaggle but you can play around with this by downloading the notebook on your machine :)

```
def img_pred(upload):
    for name, file_info in uploader.value.items():
        img = Image.open(io.BytesIO(file_info['content']))
    opencvImage = cv2.cvtColor(np.array(img), cv2.COLOR_RGB2BGR)
    img = cv2.resize(opencvImage,(150,150))
    img = img.reshape(1,150,150,3)
    p = model.predict(img)
    p = np.argmax(p,axis=1)[0]

    if p==0:
        p='Glioma Tumor'
    elif p==1:
        print('The model predicts that there is no tumor')
    elif p==2:
        p='Meningioma Tumor'
    else:
        p='Pituitary Tumor'

    if p!=1:
        print(f'The Model predicts that it is a {p}')
```

This is where you can upload the image by clicking on the **Upload** button:

```
def on_button_clicked(_):
    with out:
        clear_output()
        try:
            img_pred(uploader)

        except:
            print('No Image Uploaded/Invalid Image File')

uploader = widgets.FileUpload()
display(uploader)
button = widgets.Button(description='Predict')
out = widgets.Output()

button.on_click(on_button_clicked)
widgets.VBox([button,out])

{"model_id":"bb9f6cb74d684398a4097e76feab6d44","version_major":2,"vers
ion_minor":0}

{"model_id":"8d74353a64774dfdbff05fccca99d7d6","version_major":2,"vers
ion_minor":0}
```

After uploading the image, you can click on the **Predict** button below to make predictions:

Vision Transformers

```
pip install -U tensorflow-addons
```

```
Collecting tensorflow-addons
  Downloading tensorflow_addons-0.16.1-cp37-cp37m-
manylinux_2_12_x86_64.manylinux2010_x86_64.whl (1.1 MB)
ent already satisfied: typeguard>=2.7 in
/usr/local/lib/python3.7/dist-packages (from tensorflow-addons)
(2.7.1)
Installing collected packages: tensorflow-addons
Successfully installed tensorflow-addons-0.16.1
```

```
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import tensorflow_addons as tfa
```

```
X_train = []
y_train = []
image_size = 150
dir =
'/content/drive/MyDrive/Projects-for-Sale/Brain_Tumor_Classification/
dataset'
for i in labels:
    folderPath = os.path.join(dir,'Training',i)
    for j in tqdm(os.listdir(folderPath)):
        img = cv2.imread(os.path.join(folderPath,j))
        img = cv2.resize(img,(image_size, image_size))
        X_train.append(img)
        y_train.append(i)

for i in labels:
    folderPath = os.path.join(dir,'Testing',i)
    for j in tqdm(os.listdir(folderPath)):
        img = cv2.imread(os.path.join(folderPath,j))
        img = cv2.resize(img,(image_size,image_size))
        X_train.append(img)
        y_train.append(i)

X_train = np.array(X_train)
y_train = np.array(y_train)
```

```
100%|██████████| 276/276 [00:02<00:00, 119.91it/s]
100%|██████████| 268/268 [00:02<00:00, 114.85it/s]
```

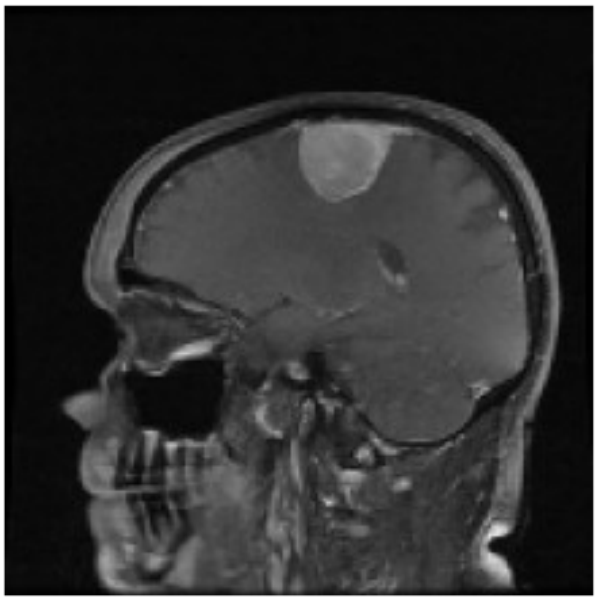
```
100%|██████████| 275/275 [00:02<00:00, 94.96it/s]
100%|██████████| 271/271 [00:03<00:00, 84.36it/s]
100%|██████████| 64/64 [00:00<00:00, 88.45it/s]
100%|██████████| 64/64 [00:00<00:00, 114.47it/s]
100%|██████████| 68/68 [00:00<00:00, 188.34it/s]
100%|██████████| 64/64 [00:00<00:00, 111.03it/s]
```

```
num_classes = 4
input_shape = (150, 150, 3)
(x_train, x_test, y_train, y_test)  =
train_test_split(X_train,y_train, test_size=0.1,random_state=101)
print(f"x_train shape: {x_train.shape} - y_train shape:
{y_train.shape}")
print(f"x_test shape: {x_test.shape} - y_test shape: {y_test.shape}")
```

```
x_train shape: (1215, 150, 150, 3) - y_train shape: (1215,)
x_test shape: (135, 150, 150, 3) - y_test shape: (135,)
```

```
data_augmentation = keras.Sequential(
    [
        layers.Normalization(),
        layers.Resizing(72, 72),
        layers.RandomFlip("horizontal"),
        layers.RandomRotation(factor=0.02),
        layers.RandomZoom(
            height_factor=0.2, width_factor=0.2
        ),
    ],
    name="data_augmentation",
)
# Compute the mean and the variance of the training data for
normalization.
data_augmentation.layers[0].adapt(x_train)

def mlp(x, hidden_units, dropout_rate):
    for units in hidden_units:
        x = layers.Dense(units, activation=tf.nn.gelu)(x)
        x = layers.Dropout(dropout_rate)(x)
    return x

import matplotlib.pyplot as plt

plt.figure(figsize=(4, 4))
image = x_train[np.random.choice(range(x_train.shape[0]))]
plt.imshow(image.astype("uint8"))
plt.axis("off")
```

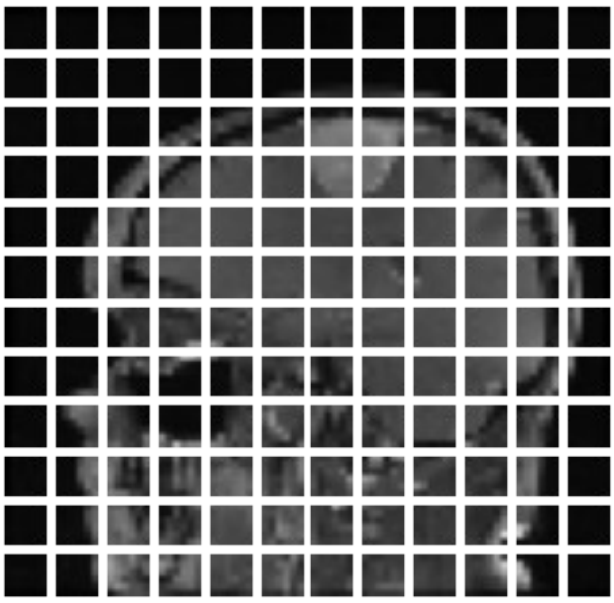
```
(-0.5, 149.5, 149.5, -0.5)
```



```
learning_rate = 0.001
weight_decay = 0.0001
batch_size = 16
num_epochs = 28
image_size = 72  # We'll resize input images to this size
patch_size = 6  # Size of the patches to be extract from the input
images
num_patches = (image_size // patch_size) ** 2
projection_dim = 64
num_heads = 4
transformer_units = [
    projection_dim * 2,
    projection_dim,
]  # Size of the transformer layers
transformer_layers = 8
mlp_head_units = [2048, 1024]  # Size of the dense layers of the final
classifier

class Patches(layers.Layer):
    def __init__(self, patch_size):
        super(Patches, self).__init__()
        self.patch_size = patch_size

    def call(self, images):
        batch_size = tf.shape(images)[0]
        patches = tf.image.extract_patches(
            images=images,
            sizes=[1, self.patch_size, self.patch_size, 1],
            strides=[1, self.patch_size, self.patch_size, 1],
            rates=[1, 1, 1, 1],
            padding="VALID",
```

```
        )
        patch_dims = patches.shape[-1]
        patches = tf.reshape(patches, [batch_size, -1, patch_dims])
        return patches

resized_image = tf.image.resize(
    tf.convert_to_tensor([image]), size=(image_size, image_size)
)
patches = Patches(patch_size)(resized_image)
print(f"Image size: {image_size} X {image_size}")
print(f"Patch size: {patch_size} X {patch_size}")
print(f"Patches per image: {patches.shape[1]}")
print(f"Elements per patch: {patches.shape[-1]}")

n = int(np.sqrt(patches.shape[1]))
plt.figure(figsize=(4, 4))
for i, patch in enumerate(patches[0]):
    ax = plt.subplot(n, n, i + 1)
    patch_img = tf.reshape(patch, (patch_size, patch_size, 3))
    plt.imshow(patch_img.numpy().astype("uint8"))
    plt.axis("off")
```

```
Image size: 72 X 72
Patch size: 6 X 6
Patches per image: 144
Elements per patch: 108
```



```
class PatchEncoder(layers.Layer):
    def __init__(self, num_patches, projection_dim):
        super(PatchEncoder, self).__init__()
```

```
        self.num_patches = num_patches
        self.projection = layers.Dense(units=projection_dim)
        self.position_embedding = layers.Embedding(
            input_dim=num_patches, output_dim=projection_dim
        )

    def call(self, patch):
        positions = tf.range(start=0, limit=self.num_patches, delta=1)
        encoded = self.projection(patch) +
self.position_embedding(positions)
        return encoded

def create_vit_classifier():
    inputs = layers.Input(shape=input_shape)
    # Augment data.
    augmented = data_augmentation(inputs)
    # Create patches.
    patches = Patches(patch_size)(augmented)
    # Encode patches.
    encoded_patches = PatchEncoder(num_patches, projection_dim)
(patches)

    # Create multiple layers of the Transformer block.
    for _ in range(transformer_layers):
        # Layer normalization 1.
        x1 = layers.LayerNormalization(epsilon=1e-6)(encoded_patches)
        # Create a multi-head attention layer.
        attention_output = layers.MultiHeadAttention(
            num_heads=num_heads, key_dim=projection_dim, dropout=0.1
        )(x1, x1)
        # Skip connection 1.
        x2 = layers.Add()([attention_output, encoded_patches])
        # Layer normalization 2.
        x3 = layers.LayerNormalization(epsilon=1e-6)(x2)
        # MLP.
        x3 = mlp(x3, hidden_units=transformer_units, dropout_rate=0.1)
        # Skip connection 2.
        encoded_patches = layers.Add()([x3, x2])

    # Create a [batch_size, projection_dim] tensor.
    representation = layers.LayerNormalization(epsilon=1e-6)
(encoded_patches)
    representation = layers.Flatten()(representation)
    representation = layers.Dropout(0.5)(representation)
    # Add MLP.
    features = mlp(representation, hidden_units=mlp_head_units,
dropout_rate=0.5)
    # Classify outputs.
    logits = layers.Dense(num_classes)(features)
    # Create the Keras model.
```

```
    model = keras.Model(inputs=inputs, outputs=logits)
    return model

!pwd

/content

def run_experiment(model):
    optimizer = tfa.optimizers.AdamW(
        learning_rate=learning_rate, weight_decay=weight_decay
    )

    model.compile(
        optimizer=optimizer,

loss=keras.losses.SparseCategoricalCrossentropy(from_logits=True),
        metrics=[
            keras.metrics.SparseCategoricalAccuracy(name="accuracy"),
            keras.metrics.SparseTopKCategoricalAccuracy(5, name="top-
5-accuracy"),
        ],
    )

    checkpoint_filepath = "/checkpoint"
    checkpoint_callback = keras.callbacks.ModelCheckpoint(
        checkpoint_filepath,
        monitor="val_accuracy",
        save_best_only=True,
        save_weights_only=True,
    )

    history = model.fit(
        x=x_train,
        y=y_train,
        batch_size=batch_size,
        epochs=num_epochs,
        validation_split=0.1,
        callbacks=[checkpoint_callback],
    )

    model.load_weights(checkpoint_filepath)
    _, accuracy, top_5_accuracy = model.evaluate(x_test, y_test)
    print(f"Test accuracy: {round(accuracy * 100, 2)}%")
    print(f"Test top 5 accuracy: {round(top_5_accuracy * 100, 2)}%")

    return history


vit_classifier = create_vit_classifier()
history = run_experiment(vit_classifier)
```

```
Epoch 1/28

---------------------------------------------------------------------------
UnimplementedError                        Traceback (most recent call last)
<ipython-input-40-e2b2386fcf33> in <module>()
     39
     40 vit_classifier = create_vit_classifier()
---> 41 history = run_experiment(vit_classifier)

<ipython-input-40-e2b2386fcf33> in run_experiment(model)
     27         epochs=num_epochs,
     28         validation_split=0.1,
---> 29         callbacks=[checkpoint_callback],
     30     )
     31

/usr/local/lib/python3.7/dist-packages/keras/utils/traceback_utils.py in error_handler(*args, **kwargs)
     65     except Exception as e:  # pylint: disable=broad-except
     66       filtered_tb = _process_traceback_frames(e.__traceback__)
---> 67       raise e.with_traceback(filtered_tb) from None
     68     finally:
     69       del filtered_tb

/usr/local/lib/python3.7/dist-packages/tensorflow/python/eager/execute.py in quick_execute(op_name, num_outputs, inputs, attrs, ctx, name)
     53     ctx.ensure_initialized()
     54     tensors = pywrap_tfe.TFE_Py_Execute(ctx._handle, device_name, op_name,
---> 55                                         inputs, attrs, num_outputs)
     56   except core._NotOkStatusException as e:
     57     if name is not None:

UnimplementedError: Graph execution error:

Detected at node 'Cast_1' defined at (most recent call last):
    File "/usr/lib/python3.7/runpy.py", line 193, in _run_module_as_main
      "__main__", mod_spec)
    File "/usr/lib/python3.7/runpy.py", line 85, in _run_code
      exec(code, run_globals)
    File "/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py", line 16, in <module>
      app.launch_new_instance()
    File "/usr/local/lib/python3.7/dist-packages/traitlets/config/application.p
```

```
y", line 846, in launch_instance
      app.start()
    File
"/usr/local/lib/python3.7/dist-packages/ipykernel/kernelapp.py", line
499, in start
      self.io_loop.start()
    File
"/usr/local/lib/python3.7/dist-packages/tornado/platform/asyncio.py",
line 132, in start
      self.asyncio_loop.run_forever()
    File "/usr/lib/python3.7/asyncio/base_events.py", line 541, in
run_forever
      self._run_once()
    File "/usr/lib/python3.7/asyncio/base_events.py", line 1786, in
_run_once
      handle._run()
    File "/usr/lib/python3.7/asyncio/events.py", line 88, in _run
      self._context.run(self._callback, *self._args)
    File
"/usr/local/lib/python3.7/dist-packages/tornado/platform/asyncio.py",
line 122, in _handle_events
      handler_func(fileobj, events)
    File
"/usr/local/lib/python3.7/dist-packages/tornado/stack_context.py",
line 300, in null_wrapper
      return fn(*args, **kwargs)
    File
"/usr/local/lib/python3.7/dist-packages/zmq/eventloop/zmqstream.py",
line 452, in _handle_events
      self._handle_recv()
    File
"/usr/local/lib/python3.7/dist-packages/zmq/eventloop/zmqstream.py",
line 481, in _handle_recv
      self._run_callback(callback, msg)
    File
"/usr/local/lib/python3.7/dist-packages/zmq/eventloop/zmqstream.py",
line 431, in _run_callback
      callback(*args, **kwargs)
    File
"/usr/local/lib/python3.7/dist-packages/tornado/stack_context.py",
line 300, in null_wrapper
      return fn(*args, **kwargs)
    File
"/usr/local/lib/python3.7/dist-packages/ipykernel/kernelbase.py", line
283, in dispatcher
      return self.dispatch_shell(stream, msg)
    File
"/usr/local/lib/python3.7/dist-packages/ipykernel/kernelbase.py", line
233, in dispatch_shell
```

```
      handler(stream, idents, msg)
    File
"/usr/local/lib/python3.7/dist-packages/ipykernel/kernelbase.py", line
399, in execute_request
      user_expressions, allow_stdin)
    File
"/usr/local/lib/python3.7/dist-packages/ipykernel/ipkernel.py", line
208, in do_execute
      res = shell.run_cell(code, store_history=store_history,
silent=silent)
    File
"/usr/local/lib/python3.7/dist-packages/ipykernel/zmqshell.py", line
537, in run_cell
      return super(ZMQInteractiveShell, self).run_cell(*args,
**kwargs)
    File
"/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.
py", line 2718, in run_cell
      interactivity=interactivity, compiler=compiler, result=result)
    File
"/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.
py", line 2822, in run_ast_nodes
      if self.run_code(code, result):
    File
"/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.
py", line 2882, in run_code
      exec(code_obj, self.user_global_ns, self.user_ns)
    File "<ipython-input-40-e2b2386fcf33>", line 41, in <module>
      history = run_experiment(vit_classifier)
    File "<ipython-input-40-e2b2386fcf33>", line 29, in run_experiment
      callbacks=[checkpoint_callback],
    File
"/usr/local/lib/python3.7/dist-packages/keras/utils/traceback_utils.py
", line 64, in error_handler
      return fn(*args, **kwargs)
    File
"/usr/local/lib/python3.7/dist-packages/keras/engine/training.py",
line 1384, in fit
      tmp_logs = self.train_function(iterator)
    File
"/usr/local/lib/python3.7/dist-packages/keras/engine/training.py",
line 1021, in train_function
      return step_function(self, iterator)
    File
"/usr/local/lib/python3.7/dist-packages/keras/engine/training.py",
line 1010, in step_function
      outputs = model.distribute_strategy.run(run_step, args=(data,))
    File
"/usr/local/lib/python3.7/dist-packages/keras/engine/training.py",
```

```
line 1000, in run_step
      outputs = model.train_step(data)
    File
"/usr/local/lib/python3.7/dist-packages/keras/engine/training.py",
line 864, in train_step
      return self.compute_metrics(x, y, y_pred, sample_weight)
    File
"/usr/local/lib/python3.7/dist-packages/keras/engine/training.py",
line 957, in compute_metrics
      self.compiled_metrics.update_state(y, y_pred, sample_weight)
    File
"/usr/local/lib/python3.7/dist-packages/keras/engine/compile_utils.py"
, line 459, in update_state
      metric_obj.update_state(y_t, y_p, sample_weight=mask)
    File
"/usr/local/lib/python3.7/dist-packages/keras/utils/metrics_utils.py",
line 70, in decorated
      update_op = update_state_fn(*args, **kwargs)
    File "/usr/local/lib/python3.7/dist-packages/keras/metrics.py",
line 178, in update_state_fn
      return ag_update_state(*args, **kwargs)
    File "/usr/local/lib/python3.7/dist-packages/keras/metrics.py",
line 720, in update_state
      y_true = tf.cast(y_true, self._dtype)
Node: 'Cast_1'
Cast string to float is not supported
       [[{{node Cast_1}}]] [Op:__inference_train_function_120093]
```