Deeper Networks for Image Classifications

Author: Norman Wagner Bright (160424285)

1. Introduction

In this paper, two deep networks are used to analyse a dataset which are ResNet and VGG16. Both models will be investigated below.
The chosen dataset is the MNIST dataset which is a collection of handwritten digits. The training and testing procedure will be analysed.In addition, a comparison will be made between the two different deep networks (ResNet and VGG16) in regards of quality of the output, train and test accuracy, and loss changing during training periods.

2. Critical analysis/Related Work

VGG (Visual Geometry Group)

Visual Geometry Group focuses on the depth in Convolutional neural networks for large-scale image recognition as presented by Karen Simonyan et al[1],
This architecture is focused on small 3x3 convolutional filters which allow an improvement in respect to current state of the art configurations. This can be done by increasing the number of weight layers to 16-19.
During training, the Input used was a 224x224 RGB image. Next, some reprocessing was done. The image is then passed through various 3x3 convolutional layers, and a 1 pixel padding is applied for 3x3 convolutional networks.
In the paper the configuration of the fully connected layers is the same in all networks. This structure differs from the one used as top entries in the ILSVRC 2012 and 2013 as this does not use large convolutional filters by doing this. Even if when stacked they equals to the top entries discussed before, they still have the advantage of making the decision function more discriminative. Another advantage is the reduction of numbers of parameters assuming that the input and output of the 3x3 layers has C channels.
The results of the classification are analysed using 2 methods, top1 and top-5 error, for the majority of the experiments. The validation set was used as a test set.

Residual Network

Residual network addresses the degradation problems which might suggest that the solver has difficulties in in approximating identity mapping by multiple layers.
This paper adopts residual learning to each stacked layers. A building block is defined by the following formula : $\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x}.$ where x and y are the input and output vectors of the considered layer, F(x,{Wi} is the residual mapping to be learned.
Two main architectures have been used in the paper which are:

- Plain network: this architecture is inspired by the VGG networks , the CNN mostly have 3x3 filters. It follows two simple rules which are: the output feature map size must have the same layers of the same number of filters and if the feature map is halved the number of features is doubled. This network uses a total of 34 layers
- Residual network: based on the plain network but with some shortcut connection which allows to turn the network into its counterpart residual version.

3. Method/Model description

In this paper, I use various deeper networks for evaluating the effectiveness of deeper CNN models for image classification on MNIST.

3.1. Model architecture

I. Vgg16 is made of several convolutional layers in our network. The input has a size of 28x28 with only a single channel. This input ~~the~~ comes into the first convolutional layer with a filter of size 64. The second layer has the same size as the input and the same number of filters. The following 2 layers have a filter size of 128. After the first 4 layers, a max-pooling is executed to reduce the size of the image by half so from now on the image size will be equals to 14. After, there are 3 convolutional layers with a filter size of 256 followed by another 3 layers of convolutional layers with a filter size of 512. Max-pooling is then executed again to reduce the size of the image from 14x14 to 7x7. This is then followed by another 3 convolutional layers with filter size of 512. A last max pooling which reduces the size to 3x3. Then a flatten is applied, an operation on a tensor reshapes the tensor to have the shape that is equal to the number of elements contained in tensor non including the batch dimension. Then 2 fully connected dense layers with 2048 units with relu activation function and a last fully connected dense layer with 10 units with softamx activation function
   All convolutional layers have a reLu activation function which ensures that the output has the same size as the input and a 3x3 kernel size with a padding equals same. The max-pooling have a pool size of 2x2 and a stride of 2x2. A summary of the model is shown below (appendix 1).

II. Resnet 30 as a structure has a deeper network compared to VGG16. This is because the residual network adds a so-called skip connection.
   The architecture developed has a total of 30 convolutional plus dense layer. They all have a 3x3 kernel size. The input is the same as the VGG which is the MNIST image. Therefore, it is 28x28x1, a single convolutional layer with a filter size of 64 a residual blocks with 64,128,256 and 512 filters and average pooling with a pool size of 4 a flatten layer and a dense layer with 10 output nodes the summary is shown below (appendix 2).

4. Experiments

This project has been implemented using the MNIST dataset and the Cifar10 dataset.

4.1. Datasets

The MNIST database[3] of handwritten digits, has a training set of 60,000 examples, and a test set of 10,000 examples, divided into 10 categories. It is a subset of a larger set available from NIST. The digits have been normalized and centred in a fixed-size image.

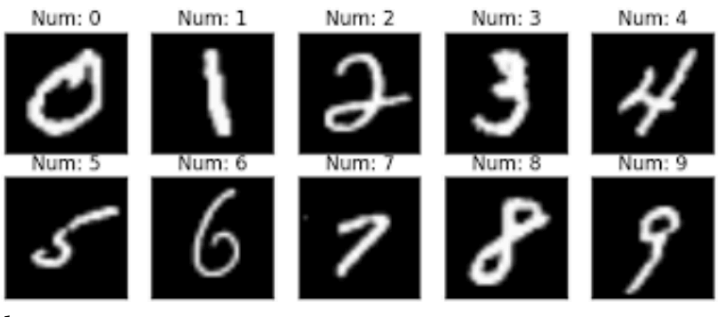

Figure 1 MNIST dataset

The second dataset[4] analysed is the Cifar10 is a dataset of 32x32 rgb images over 10 categories. It contains 50,000 train images and 10,000 test images.

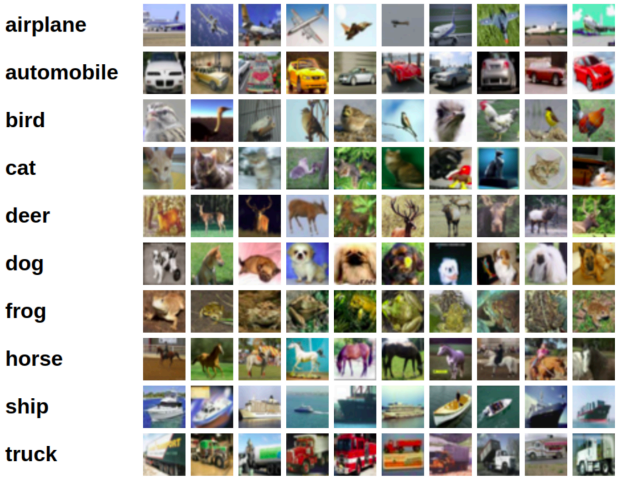

Figure 2 CIFAR10 dataset

4.2. Testing result

The learning rate used in both models is equal to 0.001. Both models used the Adam method for weight evaluation ~~this~~ which will remove the possibility of getting trapped into local optima.

The results are in the histogram shown ~~ed~~ below regarding the MNIST Dataset.

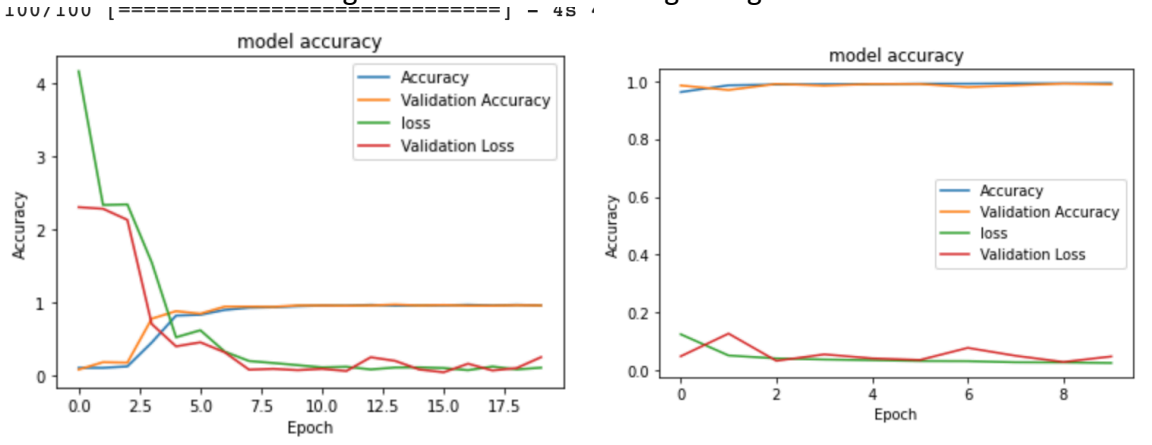

Figure 3 (a) VGG16 histogram MNIST (b) ResNet30 histogram MNIST

It is visible that at the beginning, as the VGG16 has a really high loss with time, the accuracy and validation accuracy increases. The increase occurs until it reaches an accuracy of roughly 95% while the loss gets really close to 7%, while the ResNet30 starts with an accuracy above 30% from the second epoch and a loss accuracy lower than 20%. This is because of the architecture structure and the residual network.
VGG took a time of an average of 10s per epoch. It was run on Google Colab using their GPU while ResNet took around 80 seconds per epoch.
With regards to the CIFAR10, during the VGG16 implementation, the network gets stuck in a local minimum from which cannot ~~t~~ progress. As pictured in figure5(a) where the loss is well above 200% and the accuracy never reaches a value above 10%
With regard to ResNet, the accuracy reaches the same levels as with the MNIST dataset but while it took 2 epochs for the previous one to achieve an accuracy higher than 95% with a loss lower than 10%, it then takes 20 epochs to obtain an accuracy higher than 95% while the validation accuracy Is around 85%.

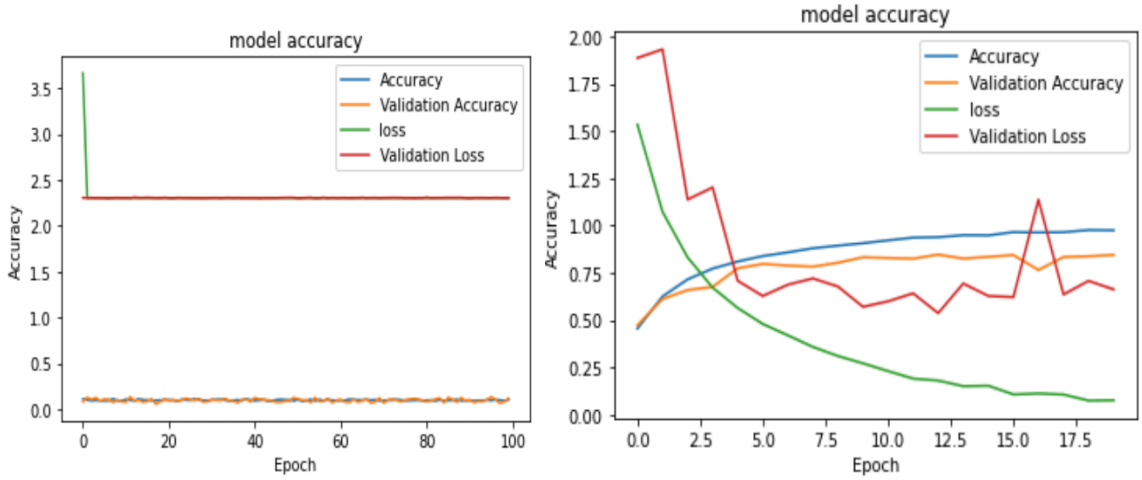

Figure 4 (a) VGG 16 histogram Cifar10 *(b) ResNet30 Histogram Cifar10*

4.3. Run time screenshots

Below there are run time screenshot of VGG and ResNet for the MNIST dataset

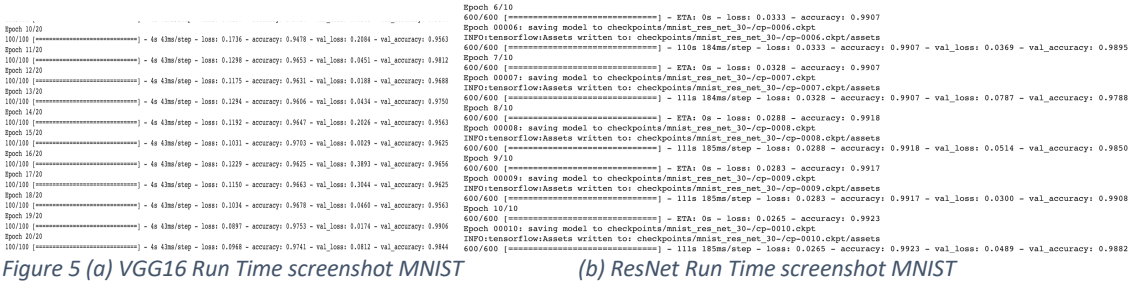

Figure 5 (a) VGG16 Run Time screenshot MNIST *(b) ResNet Run Time screenshot MNIST*

The run time screenshot in regard of the Ciifar10 dataset are shown below:

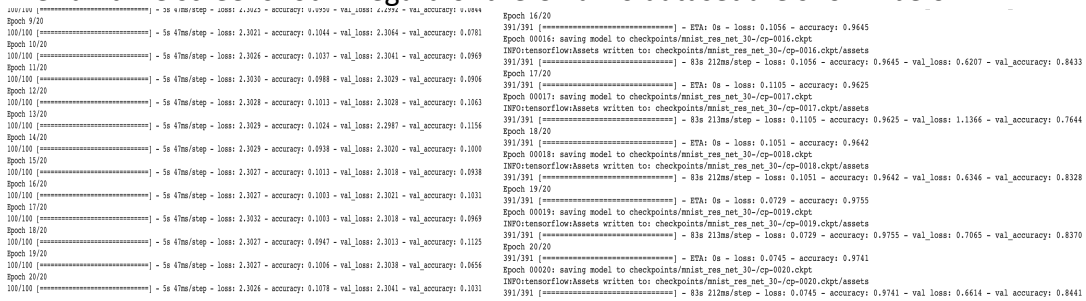

Figure 6 (a) VGG16 Run Time screenshot Cifar10 *(b) ResNet Run Time screenshot Cifar10*

4.4. Real life image testing

In this phase, we will test the model using real life images for both datasets.
All the handwritten images are resized to a 28x28 and grayscale ~~it~~ to make it easier for the network to analyse and recognise the image below. It is a sample of all the digits but only a fraction of them will be analysed.

Figure 7 Handwritten Digits[5]

Unfortunately, this wasn't achievable due to some code error. The error is due to the dimensionality of the image used to test the network as it doesn't fit the dimensionality. A resize of the image has been unsuccessfully attempted.

In regard to the Cifar10 dataset, since it spans in different classes, as not in just different numbers but each class is a different object, only 2 classes will be analysed. They are: the first class which is regarding airplanes.

Figure 8 Airplane image used for Cifar10

The VGG 16 doesn't perform well as it doesn't recognise the image. This is due to the fact that the maximum accuracy is around 10% so the network doesn't recognise the image.

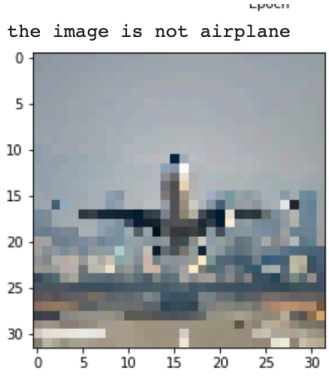

Figure 9 VGG16 image classification

The ResNet performs really well as it recognises the image. This is due to a higher accuracy around 90% as it is shown below.

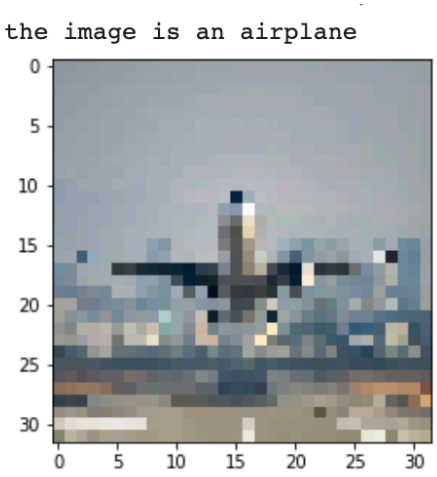

Figure 10 ResNet30 image recognition

5. Conclusions

The aim of this project was the development of a VGG and ResNet network to analyse the MNIST dataset.
It is visible as VGG16 performs really well by achieving an accuracy of above 95%, the same result is then obtained with ResNet30 but at a quicker rate as it only takes 2 epochs for it to achieve an accuracy above 95%.
The limitations of VGG16 are then highlighted with a second dataset (CIFAR10) where it is trapped in a local minima different experiments where multiple epochs have been implemented shows that the network stops learning. On the other hand ResNet30 achieve a really good accuracy with a low loss but it takes longer. This may be due to the fact that this new dataset is in RGB rather than grayscale.
In this paper the model was improved by removing 3 layers from the original which had 19 layers. This was done as a result of the reduced size of the original image from 224x224 to 28x28.
The ResNet used is ResNet30 compared to ResNet50 used in the paper.
Further experiments suggest the use of real-life handwritten digits as well as Cifar images to test the network accuracy.

References:

1) Simonyan, K. and Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
2) He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
3) Deng, L., 2012. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, *29*(6), pp.141-142.
4) Learning Multiple Layers of Features from Tiny Images, Alex Krizhevsky, 2009.
5) http://neuralnetworksanddeeplearning.com/chap1.html

Appendix 1 (VGG16 summary)

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 28, 28, 64)	640
conv2d_2 (Conv2D)	(None, 28, 28, 64)	36928
conv2d_3 (Conv2D)	(None, 28, 28, 128)	73856
conv2d_4 (Conv2D)	(None, 28, 28, 128)	147584
max_pooling2d_1 (MaxPooling2	(None, 14, 14, 128)	0
conv2d_5 (Conv2D)	(None, 14, 14, 256)	295168
conv2d_6 (Conv2D)	(None, 14, 14, 256)	590080
conv2d_7 (Conv2D)	(None, 14, 14, 256)	590080
conv2d_8 (Conv2D)	(None, 14, 14, 512)	1180160
conv2d_9 (Conv2D)	(None, 14, 14, 512)	2359808
conv2d_10 (Conv2D)	(None, 14, 14, 512)	2359808
max_pooling2d_2 (MaxPooling2	(None, 7, 7, 512)	0
conv2d_11 (Conv2D)	(None, 7, 7, 512)	2359808
conv2d_12 (Conv2D)	(None, 7, 7, 512)	2359808
conv2d_13 (Conv2D)	(None, 7, 7, 512)	2359808
max_pooling2d_3 (MaxPooling2	(None, 3, 3, 512)	0
flatten_1 (Flatten)	(None, 4608)	0
dense_1 (Dense)	(None, 2048)	9439232
dense_2 (Dense)	(None, 2048)	4196352
dense_3 (Dense)	(None, 10)	20490

Total params: 28,369,610
Trainable params: 28,369,610
Non-trainable params: 0

Appendix 2 (ResNet30 summary)

```
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #
Connected to
==================================================================================================
input_8 (InputLayer)            [(None, 32, 32, 3)]  0
__________________________________________________________________________________________________
batch_normalization_210 (BatchN (None, 32, 32, 3)    12
input_8[0][0]
__________________________________________________________________________________________________
conv2d_224 (Conv2D)             (None, 32, 32, 64)   1792
batch_normalization_210[0][0]
__________________________________________________________________________________________________
re_lu_203 (ReLU)                (None, 32, 32, 64)   0
conv2d_224[0][0]
__________________________________________________________________________________________________
batch_normalization_211 (BatchN (None, 32, 32, 64)   256
re_lu_203[0][0]
__________________________________________________________________________________________________
conv2d_225 (Conv2D)             (None, 32, 32, 64)   36928
batch_normalization_211[0][0]
__________________________________________________________________________________________________
re_lu_204 (ReLU)                (None, 32, 32, 64)   0
conv2d_225[0][0]
__________________________________________________________________________________________________
batch_normalization_212 (BatchN (None, 32, 32, 64)   256
re_lu_204[0][0]
__________________________________________________________________________________________________
conv2d_226 (Conv2D)             (None, 32, 32, 64)   36928
batch_normalization_212[0][0]
__________________________________________________________________________________________________
add_98 (Add)                    (None, 32, 32, 64)   0
batch_normalization_211[0][0]

conv2d_226[0][0]
__________________________________________________________________________________________________
re_lu_205 (ReLU)                (None, 32, 32, 64)   0
add_98[0][0]
__________________________________________________________________________________________________
batch_normalization_213 (BatchN (None, 32, 32, 64)   256
re_lu_205[0][0]
__________________________________________________________________________________________________
```

```
conv2d_227 (Conv2D)             (None, 32, 32, 64)   36928
batch_normalization_213[0][0]
__________________________________________________________________________________________________
re_lu_206 (ReLU)                (None, 32, 32, 64)   0
conv2d_227[0][0]
__________________________________________________________________________________________________
batch_normalization_214 (BatchN (None, 32, 32, 64)   256
re_lu_206[0][0]
__________________________________________________________________________________________________
conv2d_228 (Conv2D)             (None, 32, 32, 64)   36928
batch_normalization_214[0][0]
__________________________________________________________________________________________________
add_99 (Add)                    (None, 32, 32, 64)   0
batch_normalization_213[0][0]

conv2d_228[0][0]
__________________________________________________________________________________________________
re_lu_207 (ReLU)                (None, 32, 32, 64)   0
add_99[0][0]
__________________________________________________________________________________________________
batch_normalization_215 (BatchN (None, 32, 32, 64)   256
re_lu_207[0][0]
__________________________________________________________________________________________________
conv2d_229 (Conv2D)             (None, 16, 16, 128)  73856
batch_normalization_215[0][0]
__________________________________________________________________________________________________
re_lu_208 (ReLU)                (None, 16, 16, 128)  0
conv2d_229[0][0]
__________________________________________________________________________________________________
batch_normalization_216 (BatchN (None, 16, 16, 128)  512
re_lu_208[0][0]
__________________________________________________________________________________________________
conv2d_231 (Conv2D)             (None, 16, 16, 128)  8320
batch_normalization_215[0][0]
__________________________________________________________________________________________________
conv2d_230 (Conv2D)             (None, 16, 16, 128)  147584
batch_normalization_216[0][0]
__________________________________________________________________________________________________
add_100 (Add)                   (None, 16, 16, 128)  0
conv2d_231[0][0]

conv2d_230[0][0]
__________________________________________________________________________________________________
re_lu_209 (ReLU)                (None, 16, 16, 128)  0
add_100[0][0]
```

```
__________________________________________________________________________________________________
_______________________________
batch_normalization_217 (BatchN (None, 16, 16, 128)  512
re_lu_209[0][0]
__________________________________________________________________________________________________
_______________________________
conv2d_232 (Conv2D)             (None, 16, 16, 128)  147584
batch_normalization_217[0][0]
__________________________________________________________________________________________________
_______________________________
re_lu_210 (ReLU)                (None, 16, 16, 128)  0
conv2d_232[0][0]
__________________________________________________________________________________________________
_______________________________
batch_normalization_218 (BatchN (None, 16, 16, 128)  512
re_lu_210[0][0]
__________________________________________________________________________________________________
_______________________________
conv2d_233 (Conv2D)             (None, 16, 16, 128)  147584
batch_normalization_218[0][0]
__________________________________________________________________________________________________
_______________________________
add_101 (Add)                   (None, 16, 16, 128)  0
batch_normalization_217[0][0]

conv2d_233[0][0]
__________________________________________________________________________________________________
_______________________________
re_lu_211 (ReLU)                (None, 16, 16, 128)  0
add_101[0][0]
__________________________________________________________________________________________________
_______________________________
batch_normalization_219 (BatchN (None, 16, 16, 128)  512
re_lu_211[0][0]
__________________________________________________________________________________________________
_______________________________
conv2d_234 (Conv2D)             (None, 16, 16, 128)  147584
batch_normalization_219[0][0]
__________________________________________________________________________________________________
_______________________________
re_lu_212 (ReLU)                (None, 16, 16, 128)  0
conv2d_234[0][0]
__________________________________________________________________________________________________
_______________________________
batch_normalization_220 (BatchN (None, 16, 16, 128)  512
re_lu_212[0][0]
__________________________________________________________________________________________________
_______________________________
conv2d_235 (Conv2D)             (None, 16, 16, 128)  147584
batch_normalization_220[0][0]
__________________________________________________________________________________________________
_______________________________
add_102 (Add)                   (None, 16, 16, 128)  0
batch_normalization_219[0][0]

conv2d_235[0][0]
__________________________________________________________________________________________________
_______________________________
```

```
re_lu_213 (ReLU)                (None, 16, 16, 128)  0
add_102[0][0]
__________________________________________________________________________________________________
batch_normalization_221 (BatchN (None, 16, 16, 128)  512
re_lu_213[0][0]
__________________________________________________________________________________________________
conv2d_236 (Conv2D)             (None, 16, 16, 128)  147584
batch_normalization_221[0][0]
__________________________________________________________________________________________________
re_lu_214 (ReLU)                (None, 16, 16, 128)  0
conv2d_236[0][0]
__________________________________________________________________________________________________
batch_normalization_222 (BatchN (None, 16, 16, 128)  512
re_lu_214[0][0]
__________________________________________________________________________________________________
conv2d_237 (Conv2D)             (None, 16, 16, 128)  147584
batch_normalization_222[0][0]
__________________________________________________________________________________________________
add_103 (Add)                   (None, 16, 16, 128)  0
batch_normalization_221[0][0]

conv2d_237[0][0]
__________________________________________________________________________________________________
re_lu_215 (ReLU)                (None, 16, 16, 128)  0
add_103[0][0]
__________________________________________________________________________________________________
batch_normalization_223 (BatchN (None, 16, 16, 128)  512
re_lu_215[0][0]
__________________________________________________________________________________________________
conv2d_238 (Conv2D)             (None, 16, 16, 128)  147584
batch_normalization_223[0][0]
__________________________________________________________________________________________________
re_lu_216 (ReLU)                (None, 16, 16, 128)  0
conv2d_238[0][0]
__________________________________________________________________________________________________
batch_normalization_224 (BatchN (None, 16, 16, 128)  512
re_lu_216[0][0]
__________________________________________________________________________________________________
conv2d_239 (Conv2D)             (None, 16, 16, 128)  147584
batch_normalization_224[0][0]
__________________________________________________________________________________________________
add_104 (Add)                   (None, 16, 16, 128)  0
batch_normalization_223[0][0]

conv2d_239[0][0]
```

```
__________________________________________________________________________________________________
re_lu_217 (ReLU)                (None, 16, 16, 128)  0           add_104[0][0]
__________________________________________________________________________________________________
batch_normalization_225 (BatchN (None, 16, 16, 128)  512         re_lu_217[0][0]
__________________________________________________________________________________________________
conv2d_240 (Conv2D)             (None, 8, 8, 256)    295168      batch_normalization_225[0][0]
__________________________________________________________________________________________________
re_lu_218 (ReLU)                (None, 8, 8, 256)    0           conv2d_240[0][0]
__________________________________________________________________________________________________
batch_normalization_226 (BatchN (None, 8, 8, 256)    1024        re_lu_218[0][0]
__________________________________________________________________________________________________
conv2d_242 (Conv2D)             (None, 8, 8, 256)    33024       batch_normalization_225[0][0]
__________________________________________________________________________________________________
conv2d_241 (Conv2D)             (None, 8, 8, 256)    590080      batch_normalization_226[0][0]
__________________________________________________________________________________________________
add_105 (Add)                   (None, 8, 8, 256)    0           conv2d_242[0][0]

                                                                 conv2d_241[0][0]
__________________________________________________________________________________________________
re_lu_219 (ReLU)                (None, 8, 8, 256)    0           add_105[0][0]
__________________________________________________________________________________________________
batch_normalization_227 (BatchN (None, 8, 8, 256)    1024        re_lu_219[0][0]
__________________________________________________________________________________________________
conv2d_243 (Conv2D)             (None, 8, 8, 256)    590080      batch_normalization_227[0][0]
__________________________________________________________________________________________________
re_lu_220 (ReLU)                (None, 8, 8, 256)    0           conv2d_243[0][0]
__________________________________________________________________________________________________
batch_normalization_228 (BatchN (None, 8, 8, 256)    1024        re_lu_220[0][0]
__________________________________________________________________________________________________
conv2d_244 (Conv2D)             (None, 8, 8, 256)    590080      batch_normalization_228[0][0]
```

```
__________________________________________________________________________________________________
add_106 (Add)                   (None, 8, 8, 256)    0
batch_normalization_227[0][0]

conv2d_244[0][0]
__________________________________________________________________________________________________
re_lu_221 (ReLU)                (None, 8, 8, 256)    0
add_106[0][0]
__________________________________________________________________________________________________
batch_normalization_229 (BatchN (None, 8, 8, 256)    1024
re_lu_221[0][0]
__________________________________________________________________________________________________
conv2d_245 (Conv2D)             (None, 8, 8, 256)    590080
batch_normalization_229[0][0]
__________________________________________________________________________________________________
re_lu_222 (ReLU)                (None, 8, 8, 256)    0
conv2d_245[0][0]
__________________________________________________________________________________________________
batch_normalization_230 (BatchN (None, 8, 8, 256)    1024
re_lu_222[0][0]
__________________________________________________________________________________________________
conv2d_246 (Conv2D)             (None, 8, 8, 256)    590080
batch_normalization_230[0][0]
__________________________________________________________________________________________________
add_107 (Add)                   (None, 8, 8, 256)    0
batch_normalization_229[0][0]

conv2d_246[0][0]
__________________________________________________________________________________________________
re_lu_223 (ReLU)                (None, 8, 8, 256)    0
add_107[0][0]
__________________________________________________________________________________________________
batch_normalization_231 (BatchN (None, 8, 8, 256)    1024
re_lu_223[0][0]
__________________________________________________________________________________________________
conv2d_247 (Conv2D)             (None, 8, 8, 256)    590080
batch_normalization_231[0][0]
__________________________________________________________________________________________________
re_lu_224 (ReLU)                (None, 8, 8, 256)    0
conv2d_247[0][0]
__________________________________________________________________________________________________
batch_normalization_232 (BatchN (None, 8, 8, 256)    1024
re_lu_224[0][0]
__________________________________________________________________________________________________
```

```
conv2d_248 (Conv2D)             (None, 8, 8, 256)    590080
batch_normalization_232[0][0]
__________________________________________________________________________________________
_____
add_108 (Add)                   (None, 8, 8, 256)    0
batch_normalization_231[0][0]

conv2d_248[0][0]
__________________________________________________________________________________________
_____
re_lu_225 (ReLU)                (None, 8, 8, 256)    0
add_108[0][0]
__________________________________________________________________________________________
_____
batch_normalization_233 (BatchN (None, 8, 8, 256)    1024
re_lu_225[0][0]
__________________________________________________________________________________________
_____
conv2d_249 (Conv2D)             (None, 8, 8, 256)    590080
batch_normalization_233[0][0]
__________________________________________________________________________________________
_____
re_lu_226 (ReLU)                (None, 8, 8, 256)    0
conv2d_249[0][0]
__________________________________________________________________________________________
_____
batch_normalization_234 (BatchN (None, 8, 8, 256)    1024
re_lu_226[0][0]
__________________________________________________________________________________________
_____
conv2d_250 (Conv2D)             (None, 8, 8, 256)    590080
batch_normalization_234[0][0]
__________________________________________________________________________________________
_____
add_109 (Add)                   (None, 8, 8, 256)    0
batch_normalization_233[0][0]

conv2d_250[0][0]
__________________________________________________________________________________________
_____
re_lu_227 (ReLU)                (None, 8, 8, 256)    0
add_109[0][0]
__________________________________________________________________________________________
_____
batch_normalization_235 (BatchN (None, 8, 8, 256)    1024
re_lu_227[0][0]
__________________________________________________________________________________________
_____
conv2d_251 (Conv2D)             (None, 4, 4, 512)    1180160
batch_normalization_235[0][0]
__________________________________________________________________________________________
_____
re_lu_228 (ReLU)                (None, 4, 4, 512)    0
conv2d_251[0][0]
__________________________________________________________________________________________
_____
batch_normalization_236 (BatchN (None, 4, 4, 512)    2048
re_lu_228[0][0]
```

```
__________________________________________________________________________________________________
__________________________
conv2d_253 (Conv2D)             (None, 4, 4, 512)    131584
batch_normalization_235[0][0]
__________________________________________________________________________________________________
__________________________
conv2d_252 (Conv2D)             (None, 4, 4, 512)    2359808
batch_normalization_236[0][0]
__________________________________________________________________________________________________
__________________________
add_110 (Add)                   (None, 4, 4, 512)    0
conv2d_253[0][0]

conv2d_252[0][0]
__________________________________________________________________________________________________
__________________________
re_lu_229 (ReLU)                (None, 4, 4, 512)    0
add_110[0][0]
__________________________________________________________________________________________________
__________________________
batch_normalization_237 (BatchN (None, 4, 4, 512)    2048
re_lu_229[0][0]
__________________________________________________________________________________________________
__________________________
conv2d_254 (Conv2D)             (None, 4, 4, 512)    2359808
batch_normalization_237[0][0]
__________________________________________________________________________________________________
__________________________
re_lu_230 (ReLU)                (None, 4, 4, 512)    0
conv2d_254[0][0]
__________________________________________________________________________________________________
__________________________
batch_normalization_238 (BatchN (None, 4, 4, 512)    2048
re_lu_230[0][0]
__________________________________________________________________________________________________
__________________________
conv2d_255 (Conv2D)             (None, 4, 4, 512)    2359808
batch_normalization_238[0][0]
__________________________________________________________________________________________________
__________________________
add_111 (Add)                   (None, 4, 4, 512)    0
batch_normalization_237[0][0]

conv2d_255[0][0]
__________________________________________________________________________________________________
__________________________
re_lu_231 (ReLU)                (None, 4, 4, 512)    0
add_111[0][0]
__________________________________________________________________________________________________
__________________________
batch_normalization_239 (BatchN (None, 4, 4, 512)    2048
re_lu_231[0][0]
__________________________________________________________________________________________________
__________________________
average_pooling2d_7 (AveragePoo (None, 1, 1, 512)    0
batch_normalization_239[0][0]
__________________________________________________________________________________________________
__________________________
```

```
flatten_7 (Flatten)             (None, 512)          0
average_pooling2d_7[0][0]
__________________________________________________________________________________
_____________________________
dense_7 (Dense)                 (None, 10)           5130
flatten_7[0][0]
==================================================================================
============================
Total params: 15,619,990
Trainable params: 15,607,568
Non-trainable params: 12,422
```