Deep Learning Assignment

880008-M-6 / Master DSS

Wafa Mohamed

Student Number: 2097778

Group: 10

Date: 09-10-2023

Chapter 1: Experiment Description

The dataset is highly imbalanced, with roughly 67% of images belonging to the 'Melanocytic nevi' group, (Figure 1), and 3 categories accounting for around 90% of the images. To maintain data integrity, we first split the dataset into training, validation, and testing, resulting in a sample size of 6409, 1603, and 200, respectively. We normalized each dataset separately to ensure pixels values ranged from 0 to 1. This normalization was based on the minimum and maximum values of the training set.

We implemented the baseline model as specified, we only set the padding to 'same' to maintain input-output size consistency. The activation function for all models' output layers is softmax as this is a multiclass problem. The softmax maximizes the probability of one class and ensures the classified image belongs to one of the categories each time. The loss function we used is the categorical cross entropy (multiclass), optimization was performed using the validation set.

For the enhanced model, we experimented with various strategies. Firstly, we fine-tuned different hyperparameters (based on the baseline model), including dropout, the number of units, (additional) layers for convolutional, max pooling and dropout, and different optimizers. The minimum value for the dropout is set to 0 and the maximum is set to 0.3 with a step size of 0.1. The minimum number of (additional) layers is set to 0 and the maximum number of layers is set to 3. The minimum units are set to 8 and the maximum units are set to 64 with a step size of 7. For the optimizers, we fine-tuned Adam, Adagrad and RmsProp. For the hyperparameter tuning we used RandomizedSearchCV. After extensive tuning, we found the following combination of parameters to be the most effective {'units': 43, 'num_layers': 0, 'dropout': 0.2, 'optimizer': 'adam', 'dropout_1': 0.4, 'dropout_2': 0.30}. The new model has 4.6 million trainable parameters, nearly 4 times more than the baseline model (Figure 2).

Secondly, we applied data augmentation to the baseline model using 'ImageDataGenerator' without any further fine-tuning. We set the rotation_range to 75, width_shift_range = 0.3, height_shift_range = 0.3, shear_range = 0.2, zoom_range = 0.2 and horizontal_flip = True. Finally, we build a model that combines data augmentation with fining tuning.

For the Transfer Learning model, we explored two models: VGG16 and ResNet50. For both models we added one dense layer (128 filters) and output layers (dense layer). The VGG16 and ResNet50 contain 15,502,151 and 28,831,623 trainable parameters respectively (Appendix 1 and 2).

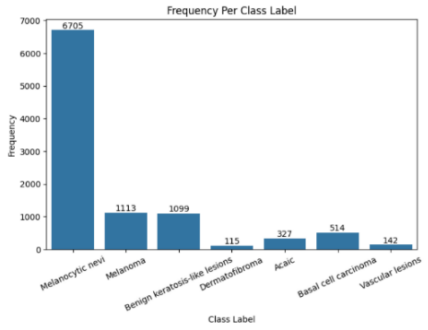


Figure 1: Class distribution

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_2 (Conv2D)            (None, 150, 120, 43)      1204

conv2d_3 (Conv2D)            (None, 150, 120, 32)      12416

max_pooling2d_1 (MaxPooling  (None, 75, 60, 32)        0
2D)

dropout_1 (Dropout)          (None, 75, 60, 32)        0

flatten_1 (Flatten)          (None, 144000)            0

dense_3 (Dense)              (None, 32)                4608032

dense_4 (Dense)              (None, 32)                1056

dense_5 (Dense)              (None, 7)                 231

=================================================================
Total params: 4,622,939
Trainable params: 4,622,939
Non-trainable params: 0
```

Figure 2: Model summary (fine-tuned model)

Chapter 2: Model Performances

BASELINE MODEL As seen in the Figure 3.A; the training accuracy increases in every epoch. The validation accuracy peaks at the fifth epoch, then fluctuates. The training loss steadily decreases, but the validation varies. From the graph, it can be observed that the model performs well on training data, but it does not perform well on the validation data, which means that the model is overfitting (high gap). The test accuracy and the test loss are 70% and 0.8 respectively. As observed in the Figure 4.G, this

model only classified 1349 out of 2003 correctly on the test data, with two classes (Acais and Dermatofibroma) misclassifying all images. The model exhibits a low mean sensitivity score (0.25, range 0-0.92) and high mean specificity score (0.92, range 0.55-1), indicating poor performance on minority classes and a bias toward the majority class. In a medical context, sensitivity is more crucial, and achieving a smaller gap between sensitivity and specificity is desirable.

The macro average AUC score for the validation data is 0.58, the same as the test data (Figure 5.K), and the micro average AUC score for the validation and test data is 0.8 and 0.81 respectively (One vs. rest methodology). Giving the severe imbalance in this medical dataset set, the preferred metric is the macro-average score (AUC or F1). The macro-average AUC metric assesses the model's accuracy for individual classes independently, giving equal importance to all classes in the dataset. The macro-ROC curve results of this model indicate inadequate discrimination for most classes, with six out of seven classes having AUC scores below 0.7 in both validation and test data (Figure 5.K)

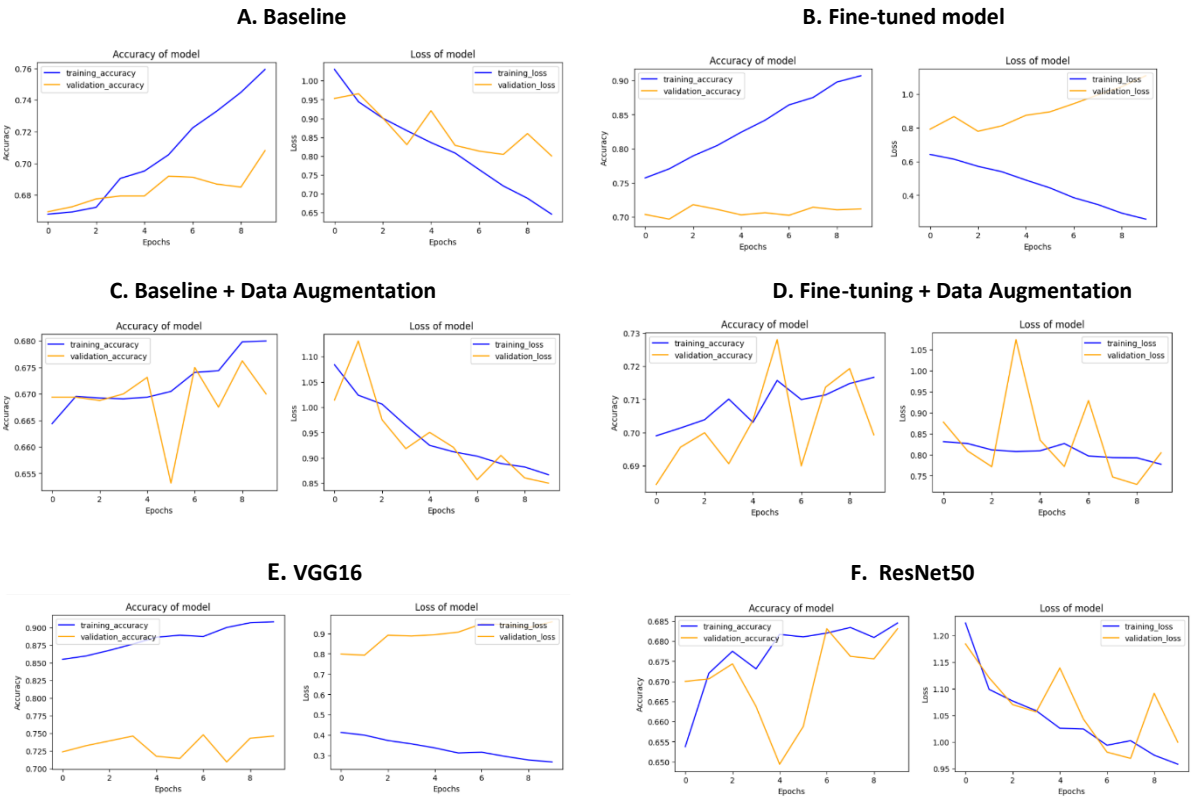


Figure 3: Accuracy and loss plots

DATA AUGMENTATION Data augmentation improved model performance compared to the baseline by reducing the gap between training and validation accuracy (Figure 3.C), this shows data augmentation helps to mitigate overfitting. Test accuracy and loss (68% and 0.98, same as validation) demonstrate its generalization ability. While data augmentation made the baseline model more resistant to overfitting, the sensitivity and specificity scores dropped to 0.17 and 0.89, respectively, compared to the baseline.

FINE-TUNED MODEL As depicted in Figure 3.B the training accuracy increases, whereas the validation accuracy stabilized after epoch 2. However, there is a notable wider gap between training accuracy and validation accuracy/loss compared to the baseline model. This increased gap may be due to the enhanced model's higher capacity, with four times more parameters than the baseline, making it more prone to overfitting, because the model is capturing unwanted noise. Interestingly, despite its susceptibility to overfitting, this model performs better on minority classes compared to the baseline

(Figure 4.H). Mean sensitivity is 0.36, an improvement over the baseline's 0.25. Moreover, the macro-average ROC curve also improves to 0.62 for both validation and test data (Figure 5.L). Combining fine-tuning with data augmentations, as shown in Figure 3.D results in a model that is more resistant to overfitting. This enhanced model with data augmentations achieves a sensitivity score of 0.34 (in the range of 0-0.79) and a specificity score of 0.93 (in the range of 0.67-1).

TRANSFER LEARNING MODELS Both the VGG16 and ResNet50 models demonstrate overfitting, with VGG16 exhibiting a larger training-validation accuracy gap (Figure 3.E and 3.F). However, VGG16 somewhat outperforms the baseline model, baseline model with data augmentation, fine-tuned model, and fine-tuned model with data augmentation on both the validation and test dataset. The validation accuracy is 73% and the validation loss is 0.89. The test accuracy of 76% is slightly better than the validation accuracy, while the test loss remains unchanged. The micro average and macro average are 0.84 and 0.67, for the validation set and 0.85 and 0.68 for the test set. This model has a sensitivity score of 0.44 (range 0.04-0.93)) and a macro-average ROC curve of 0.68, outperforming all other models. ResNet50, on the other hand, achieves a validation accuracy of 67% and a validation loss of 1.1. The test accuracy and loss only show slight changes, reaching 68% and 1, respectively. The macro-average ROC curve and sensitivity scores are respectively 0.53 and 0.22. Interestingly, all models exhibit lower sensitivity compared to specificity, implying that the classification of true positive classes is more challenging than that of true negative classes.

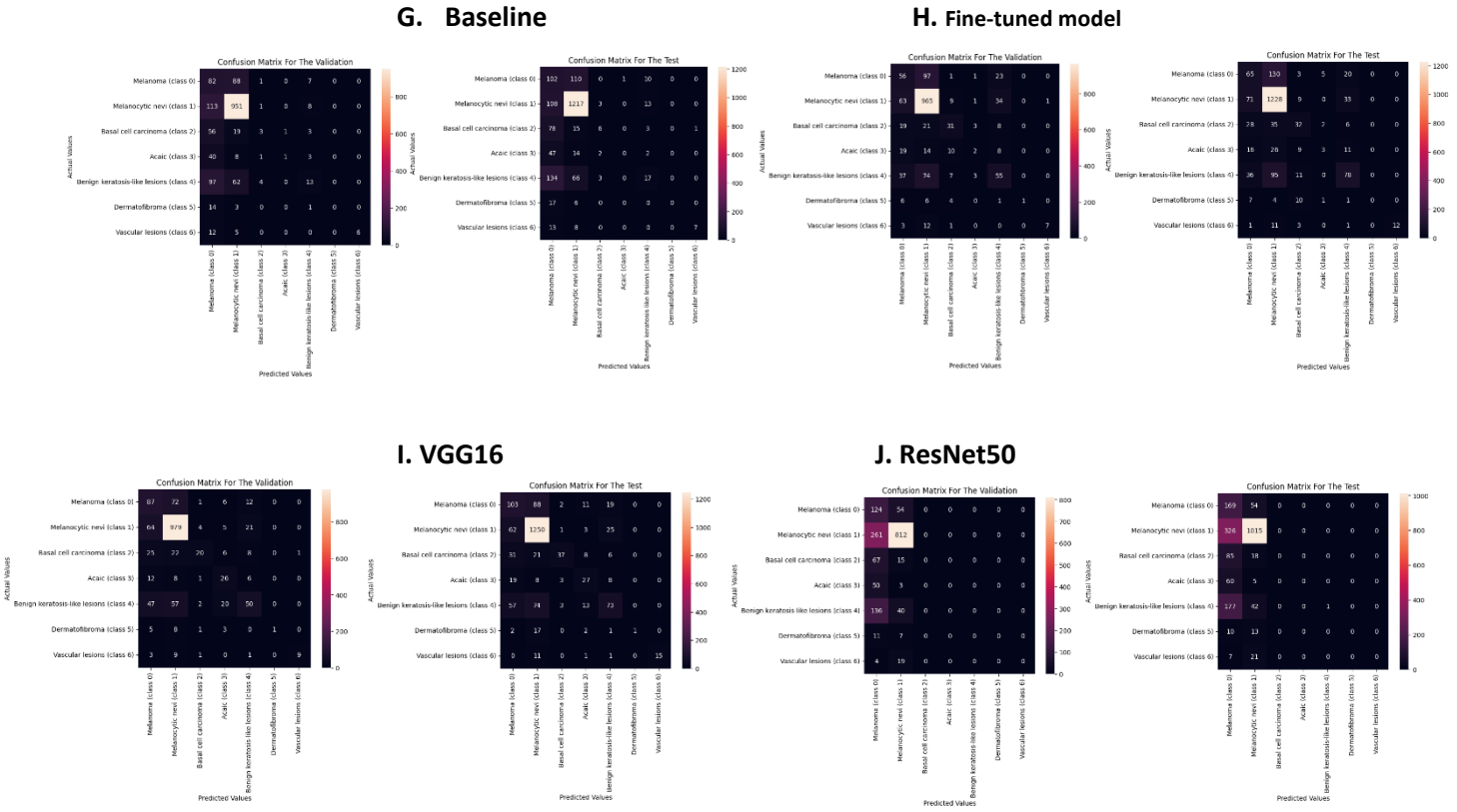


Figure 4: Confusion Matrix plots

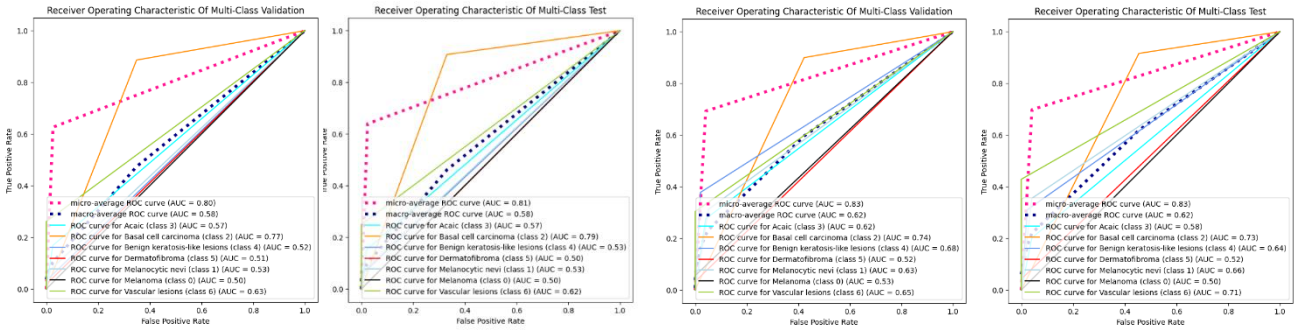
Chapter 2: Literature

CNN-based models employ strategies for performance improvement. Junaide et al. (2021) improved CNN performance by using dropout layers. Dropout prevents overfitting by randomly setting some neurons to zero with a specified probability. These neurons are then deleted from the architecture

during the training. The authors found that adding dropout after the first convolutional layer and after flattening improved results. Loffe et al (2015). used batch normalization to speed up CNN training and reduce overfitting. Batch normalization ensures neurons have a mean of zero and unit variance, increasing accuracy while reducing the need for dropout layers and training steps. Data augmentation is a popular regularization method for CNN Shorten et al (2019). Unlike other techniques like batch normalization, dropout, and transfer learning, data augmentation modifies the training dataset's size to extract additional information from the original dataset, by applying various augmentation techniques encompassing rotating, flipping, cropping, color space and translation. Due to this.

Gülgün et al (2020). introduced a hybrid CNN model for medical image classification. The authors used CNN consisting of a convolutional layer with a relu activation function, followed by a max pooling layer (2X2), repeated three times. Subsequently, a fully connected layer with sigmoid function and dropout was applied to capture the low, medium, and high-level image feature. Data augmentation was also incorporated to enhance the model performance. After feature extraction using the CNN model, the Random Forest of Support Vactor Machine algorithm was used as a different classifier to classify images. The difference between this hybrid classifier and a CNN model is that after the features of the model are extracted, the SVM or RF is applied as a different classifier instead of an artificial neural network. The CNN model with RF reached an accuracy of 100 % for two different datasets. Goumiri et al. (2023) developed a hybrid model composed of CNN and Hidden Markov Chains (HMC). Like previous hybrid CNN models, it uses CNN model consisting of four blocks of convolutions (kernel size of 3X3, filter size of 16, 32, 32, 64), max-pooling (2X2) and dropout layers (0.5) for feature extraction and HMC is used like RF and SVM for image classification. Results showed that this CNN-HMC hybrid models surpassed the standard CNN, with accuracies ranging from 81.63% to 92.5%, while the CNN achieved only 71% accuracy.

K. Baseline **L. Fine-tuned model**



M. VGG16 **N. ResNet50**

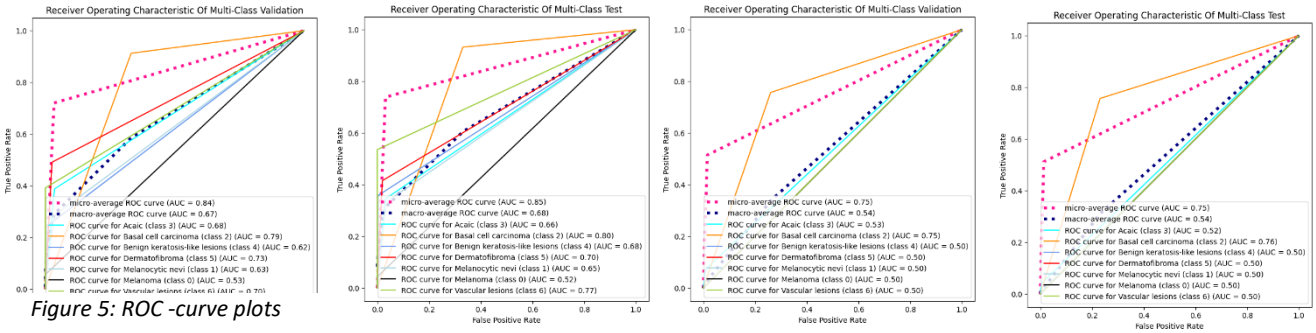


Figure 5: ROC -curve plots

Reference

Goumiri, S., Benboudjema, D. & Pieczynski, W. A new hybrid model of convolutional neural networks and hidden Markov chains for image classification. Neural Comput & Applic 35, 17987–18002 (2023). https://doi.org/10.1007/s00521-023-08644-4

A. Junaidi, N. A. Ferani Tanjung, S. Wijayanto, J. Lasama and A. R. Iskandar, "Overfitting Problem in Images Classification for Egg Incubator Using Convolutional Neural Network," 2021 9th International Conference on Cyber and IT Service Management (CITSM), Bengkulu, Indonesia, 2021, pp. 1-7, doi: 10.1109/CITSM52892.2021.9588815

S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in Proceedings of the International Conference on Machine Learning (ICML), 2015, https://doi.org/10.48550/arXiv.1502.03167

Shorten, C., Khoshgoftaar, T.M. A survey on Image Data Augmentation for Deep Learning. J Big Data 6, 60 (2019). https://doi.org/10.1186/s40537-019-0197-0

O.D. Gülgün and H. Erol, "Medical Image Classification with Hybrid Convolutional Neural Network Models," Journal of Computer Science and Technologies, vol. 1, no. 1, pp. 28-41, 2020.

Appendix

Appendix 1

Model: "VGG6"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 150, 120, 3)]	0
block1_conv1 (Conv2D)	(None, 150, 120, 64)	1792
block1_conv2 (Conv2D)	(None, 150, 120, 64)	36928
block1_pool (MaxPooling2D)	(None, 75, 60, 64)	0
block2_conv1 (Conv2D)	(None, 75, 60, 128)	73856
block2_conv2 (Conv2D)	(None, 75, 60, 128)	147584
block2_pool (MaxPooling2D)	(None, 37, 30, 128)	0
block3_conv1 (Conv2D)	(None, 37, 30, 256)	295168
block3_conv2 (Conv2D)	(None, 37, 30, 256)	590080
block3_conv3 (Conv2D)	(None, 37, 30, 256)	590080
block3_pool (MaxPooling2D)	(None, 18, 15, 256)	0
block4_conv1 (Conv2D)	(None, 18, 15, 512)	1180160
block4_conv2 (Conv2D)	(None, 18, 15, 512)	2359808
block4_conv3 (Conv2D)	(None, 18, 15, 512)	2359808
block4_pool (MaxPooling2D)	(None, 9, 7, 512)	0
block5_conv1 (Conv2D)	(None, 9, 7, 512)	2359808
block5_conv2 (Conv2D)	(None, 9, 7, 512)	2359808
block5_conv3 (Conv2D)	(None, 9, 7, 512)	2359808
block5_pool (MaxPooling2D)	(None, 4, 3, 512)	0
flatten_2 (Flatten)	(None, 6144)	0
dense_6 (Dense)	(None, 128)	786560
dense_7 (Dense)	(None, 7)	903

Total params: 15,502,151
Trainable params: 787,463
Non-trainable params: 14,714,688

Appendix 2 ResNet50

Model: "model"

```
__________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
==================================================================================================
 input_1 (InputLayer)        [(None, 150, 120, 3  0           []
                             )]

 conv1_pad (ZeroPadding2D)      (None, 156, 126, 3)  0           ['input_1[0][0]']

 conv1_conv (Conv2D)            (None, 75, 60, 64)   9472        ['conv1_pad[0][0]']

 conv1_bn (BatchNormalization)  (None, 75, 60, 64)   256         ['conv1_conv[0][0]']

 conv1_relu (Activation)        (None, 75, 60, 64)   0           ['conv1_bn[0][0]']

 pool1_pad (ZeroPadding2D)      (None, 77, 62, 64)   0           ['conv1_relu[0][0]']

 pool1_pool (MaxPooling2D)      (None, 38, 30, 64)   0           ['pool1_pad[0][0]']

 conv2_block1_1_conv (Conv2D)   (None, 38, 30, 64)   4160        ['pool1_pool[0][0]']

 conv2_block1_1_bn (BatchNormal  (None, 38, 30, 64)  256         ['conv2_block1_1_conv[0][0]']
 ization)

 conv2_block1_1_relu (Activatio  (None, 38, 30, 64)  0           ['conv2_block1_1_bn[0][0]']
 n)

 conv2_block1_2_conv (Conv2D)   (None, 38, 30, 64)   36928       ['conv2_block1_1_relu[0][0]']

 conv2_block1_2_bn (BatchNormal  (None, 38, 30, 64)  256         ['conv2_block1_2_conv[0][0]']
 ization)

 conv2_block1_2_relu (Activatio  (None, 38, 30, 64)  0           ['conv2_block1_2_bn[0][0]']
 n)

 conv2_block1_0_conv (Conv2D)   (None, 38, 30, 256)  16640       ['pool1_pool[0][0]']

 conv2_block1_3_conv (Conv2D)   (None, 38, 30, 256)  16640       ['conv2_block1_2_relu[0][0]']

 conv2_block1_0_bn (BatchNormal  (None, 38, 30, 256)  1024       ['conv2_block1_0_conv[0][0]']
 ization)

 conv2_block1_3_bn (BatchNormal  (None, 38, 30, 256)  1024       ['conv2_block1_3_conv[0][0]']
 ization)

 conv2_block1_add (Add)         (None, 38, 30, 256)  0           ['conv2_block1_0_bn[0][0]',
                                                                  'conv2_block1_3_bn[0][0]']

 conv2_block1_out (Activation)  (None, 38, 30, 256)  0           ['conv2_block1_add[0][0]']

 conv2_block2_1_conv (Conv2D)   (None, 38, 30, 64)   16448       ['conv2_block1_out[0][0]']

 conv2_block2_1_bn (BatchNormal  (None, 38, 30, 64)  256         ['conv2_block2_1_conv[0][0]']
 ization)

 conv2_block2_1_relu (Activatio  (None, 38, 30, 64)  0           ['conv2_block2_1_bn[0][0]']
 n)
```

conv2_block2_2_conv (Conv2D) (None, 38, 30, 64) 36928 ['conv2_block2_1_relu[0][0]']

conv2_block2_2_bn (BatchNormalization) (None, 38, 30, 64) 256 ['conv2_block2_2_conv[0][0]']

conv2_block2_2_relu (Activation) (None, 38, 30, 64) 0 ['conv2_block2_2_bn[0][0]']

conv2_block2_3_conv (Conv2D) (None, 38, 30, 256) 16640 ['conv2_block2_2_relu[0][0]']

conv2_block2_3_bn (BatchNormalization) (None, 38, 30, 256) 1024 ['conv2_block2_3_conv[0][0]']

conv2_block2_add (Add) (None, 38, 30, 256) 0 ['conv2_block1_out[0][0]', 'conv2_block2_3_bn[0][0]']

conv2_block2_out (Activation) (None, 38, 30, 256) 0 ['conv2_block2_add[0][0]']

conv2_block3_1_conv (Conv2D) (None, 38, 30, 64) 16448 ['conv2_block2_out[0][0]']

conv2_block3_1_bn (BatchNormalization) (None, 38, 30, 64) 256 ['conv2_block3_1_conv[0][0]']

conv2_block3_1_relu (Activation) (None, 38, 30, 64) 0 ['conv2_block3_1_bn[0][0]']

conv2_block3_2_conv (Conv2D) (None, 38, 30, 64) 36928 ['conv2_block3_1_relu[0][0]']

conv2_block3_2_bn (BatchNormalization) (None, 38, 30, 64) 256 ['conv2_block3_2_conv[0][0]']

conv2_block3_2_relu (Activation) (None, 38, 30, 64) 0 ['conv2_block3_2_bn[0][0]']

conv2_block3_3_conv (Conv2D) (None, 38, 30, 256) 16640 ['conv2_block3_2_relu[0][0]']

conv2_block3_3_bn (BatchNormalization) (None, 38, 30, 256) 1024 ['conv2_block3_3_conv[0][0]']

conv2_block3_add (Add) (None, 38, 30, 256) 0 ['conv2_block2_out[0][0]', 'conv2_block3_3_bn[0][0]']

conv2_block3_out (Activation) (None, 38, 30, 256) 0 ['conv2_block3_add[0][0]']

conv3_block1_1_conv (Conv2D) (None, 19, 15, 128) 32896 ['conv2_block3_out[0][0]']

conv3_block1_1_bn (BatchNormalization) (None, 19, 15, 128) 512 ['conv3_block1_1_conv[0][0]']

conv3_block1_1_relu (Activation) (None, 19, 15, 128) 0 ['conv3_block1_1_bn[0][0]']

conv3_block1_2_conv (Conv2D) (None, 19, 15, 128) 147584 ['conv3_block1_1_relu[0][0]']

conv3_block1_2_bn (BatchNormalization) (None, 19, 15, 128) 512 ['conv3_block1_2_conv[0][0]']

conv3_block1_2_relu (Activatio (None, 19, 15, 128) 0 ['conv3_block1_2_bn[0][0]']
n)

conv3_block1_0_conv (Conv2D) (None, 19, 15, 512) 131584 ['conv2_block3_out[0][0]']

conv3_block1_3_conv (Conv2D) (None, 19, 15, 512) 66048 ['conv3_block1_2_relu[0][0]']

conv3_block1_0_bn (BatchNormal (None, 19, 15, 512) 2048 ['conv3_block1_0_conv[0][0]']
ization)

conv3_block1_3_bn (BatchNormal (None, 19, 15, 512) 2048 ['conv3_block1_3_conv[0][0]']
ization)

conv3_block1_add (Add) (None, 19, 15, 512) 0 ['conv3_block1_0_bn[0][0]',
'conv3_block1_3_bn[0][0]']

conv3_block1_out (Activation) (None, 19, 15, 512) 0 ['conv3_block1_add[0][0]']

conv3_block2_1_conv (Conv2D) (None, 19, 15, 128) 65664 ['conv3_block1_out[0][0]']

conv3_block2_1_bn (BatchNormal (None, 19, 15, 128) 512 ['conv3_block2_1_conv[0][0]']
ization)

conv3_block2_1_relu (Activatio (None, 19, 15, 128) 0 ['conv3_block2_1_bn[0][0]']
n)

conv3_block2_2_conv (Conv2D) (None, 19, 15, 128) 147584 ['conv3_block2_1_relu[0][0]']

conv3_block2_2_bn (BatchNormal (None, 19, 15, 128) 512 ['conv3_block2_2_conv[0][0]']
ization)

conv3_block2_2_relu (Activatio (None, 19, 15, 128) 0 ['conv3_block2_2_bn[0][0]']
n)

conv3_block2_3_conv (Conv2D) (None, 19, 15, 512) 66048 ['conv3_block2_2_relu[0][0]']

conv3_block2_3_bn (BatchNormal (None, 19, 15, 512) 2048 ['conv3_block2_3_conv[0][0]']
ization)

conv3_block2_add (Add) (None, 19, 15, 512) 0 ['conv3_block1_out[0][0]',
'conv3_block2_3_bn[0][0]']

conv3_block2_out (Activation) (None, 19, 15, 512) 0 ['conv3_block2_add[0][0]']

conv3_block3_1_conv (Conv2D) (None, 19, 15, 128) 65664 ['conv3_block2_out[0][0]']

conv3_block3_1_bn (BatchNormal (None, 19, 15, 128) 512 ['conv3_block3_1_conv[0][0]']
ization)

conv3_block3_1_relu (Activatio (None, 19, 15, 128) 0 ['conv3_block3_1_bn[0][0]']
n)

conv3_block3_2_conv (Conv2D) (None, 19, 15, 128) 147584 ['conv3_block3_1_relu[0][0]']

conv3_block3_2_bn (BatchNormal (None, 19, 15, 128) 512 ['conv3_block3_2_conv[0][0]']
ization)

conv3_block3_2_relu (Activatio (None, 19, 15, 128) 0 ['conv3_block3_2_bn[0][0]']

n)

conv3_block3_3_conv (Conv2D)   (None, 19, 15, 512) 66048   ['conv3_block3_2_relu[0][0]']

conv3_block3_3_bn (BatchNormal (None, 19, 15, 512) 2048   ['conv3_block3_3_conv[0][0]']
ization)

conv3_block3_add (Add)   (None, 19, 15, 512) 0   ['conv3_block2_out[0][0]',
                                                                'conv3_block3_3_bn[0][0]']

conv3_block3_out (Activation) (None, 19, 15, 512) 0   ['conv3_block3_add[0][0]']

conv3_block4_1_conv (Conv2D)   (None, 19, 15, 128) 65664   ['conv3_block3_out[0][0]']

conv3_block4_1_bn (BatchNormal (None, 19, 15, 128) 512   ['conv3_block4_1_conv[0][0]']
ization)

conv3_block4_1_relu (Activatio (None, 19, 15, 128) 0   ['conv3_block4_1_bn[0][0]']
n)

conv3_block4_2_conv (Conv2D)   (None, 19, 15, 128) 147584   ['conv3_block4_1_relu[0][0]']

conv3_block4_2_bn (BatchNormal (None, 19, 15, 128) 512   ['conv3_block4_2_conv[0][0]']
ization)

conv3_block4_2_relu (Activatio (None, 19, 15, 128) 0   ['conv3_block4_2_bn[0][0]']
n)

conv3_block4_3_conv (Conv2D)   (None, 19, 15, 512) 66048   ['conv3_block4_2_relu[0][0]']

conv3_block4_3_bn (BatchNormal (None, 19, 15, 512) 2048   ['conv3_block4_3_conv[0][0]']
ization)

conv3_block4_add (Add)   (None, 19, 15, 512) 0   ['conv3_block3_out[0][0]',
                                                                'conv3_block4_3_bn[0][0]']

conv3_block4_out (Activation) (None, 19, 15, 512) 0   ['conv3_block4_add[0][0]']

conv4_block1_1_conv (Conv2D)   (None, 10, 8, 256)   131328   ['conv3_block4_out[0][0]']

conv4_block1_1_bn (BatchNormal (None, 10, 8, 256) 1024   ['conv4_block1_1_conv[0][0]']
ization)

conv4_block1_1_relu (Activatio (None, 10, 8, 256) 0   ['conv4_block1_1_bn[0][0]']
n)

conv4_block1_2_conv (Conv2D)   (None, 10, 8, 256)   590080   ['conv4_block1_1_relu[0][0]']

conv4_block1_2_bn (BatchNormal (None, 10, 8, 256) 1024   ['conv4_block1_2_conv[0][0]']
ization)

conv4_block1_2_relu (Activatio (None, 10, 8, 256) 0   ['conv4_block1_2_bn[0][0]']
n)

conv4_block1_0_conv (Conv2D)   (None, 10, 8, 1024) 525312   ['conv3_block4_out[0][0]']

conv4_block1_3_conv (Conv2D)   (None, 10, 8, 1024) 263168   ['conv4_block1_2_relu[0][0]']

conv4_block1_0_bn (BatchNormal (None, 10, 8, 1024) 4096   ['conv4_block1_0_conv[0][0]']
ization)

conv4_block1_3_bn (BatchNormal (None, 10, 8, 1024) 4096   ['conv4_block1_3_conv[0][0]']
ization)

conv4_block1_add (Add)   (None, 10, 8, 1024) 0   ['conv4_block1_0_bn[0][0]',
                                                                'conv4_block1_3_bn[0][0]']

conv4_block1_out (Activation) (None, 10, 8, 1024) 0   ['conv4_block1_add[0][0]']

conv4_block2_1_conv (Conv2D)   (None, 10, 8, 256)   262400   ['conv4_block1_out[0][0]']

conv4_block2_1_bn (BatchNormal (None, 10, 8, 256) 1024   ['conv4_block2_1_conv[0][0]']
ization)

conv4_block2_1_relu (Activatio (None, 10, 8, 256) 0   ['conv4_block2_1_bn[0][0]']
n)

conv4_block2_2_conv (Conv2D)   (None, 10, 8, 256)   590080   ['conv4_block2_1_relu[0][0]']

conv4_block2_2_bn (BatchNormal (None, 10, 8, 256) 1024   ['conv4_block2_2_conv[0][0]']
ization)

conv4_block2_2_relu (Activatio (None, 10, 8, 256) 0   ['conv4_block2_2_bn[0][0]']
n)

conv4_block2_3_conv (Conv2D)   (None, 10, 8, 1024) 263168   ['conv4_block2_2_relu[0][0]']

conv4_block2_3_bn (BatchNormal (None, 10, 8, 1024) 4096   ['conv4_block2_3_conv[0][0]']
ization)

conv4_block2_add (Add)   (None, 10, 8, 1024) 0   ['conv4_block1_out[0][0]',
                                                                'conv4_block2_3_bn[0][0]']

conv4_block2_out (Activation) (None, 10, 8, 1024) 0   ['conv4_block2_add[0][0]']

conv4_block3_1_conv (Conv2D)   (None, 10, 8, 256)   262400   ['conv4_block2_out[0][0]']

conv4_block3_1_bn (BatchNormal (None, 10, 8, 256) 1024   ['conv4_block3_1_conv[0][0]']
ization)

conv4_block3_1_relu (Activatio (None, 10, 8, 256) 0   ['conv4_block3_1_bn[0][0]']
n)

conv4_block3_2_conv (Conv2D)   (None, 10, 8, 256)   590080   ['conv4_block3_1_relu[0][0]']

conv4_block3_2_bn (BatchNormal (None, 10, 8, 256) 1024   ['conv4_block3_2_conv[0][0]']
ization)

conv4_block3_2_relu (Activatio (None, 10, 8, 256) 0   ['conv4_block3_2_bn[0][0]']
n)

conv4_block3_3_conv (Conv2D)   (None, 10, 8, 1024) 263168   ['conv4_block3_2_relu[0][0]']

conv4_block3_3_bn (BatchNormal (None, 10, 8, 1024) 4096   ['conv4_block3_3_conv[0][0]']
ization)

```
conv4_block3_add (Add)        (None, 10, 8, 1024)  0          ['conv4_block2_out[0][0]',
                                                               'conv4_block3_3_bn[0][0]']

conv4_block3_out (Activation)  (None, 10, 8, 1024)  0         ['conv4_block3_add[0][0]']

conv4_block4_1_conv (Conv2D)   (None, 10, 8, 256)   262400    ['conv4_block3_out[0][0]']

conv4_block4_1_bn (BatchNormal  (None, 10, 8, 256)  1024       ['conv4_block4_1_conv[0][0]']
ization)

conv4_block4_1_relu (Activatio  (None, 10, 8, 256)  0          ['conv4_block4_1_bn[0][0]']
n)

conv4_block4_2_conv (Conv2D)   (None, 10, 8, 256)   590080    ['conv4_block4_1_relu[0][0]']

conv4_block4_2_bn (BatchNormal  (None, 10, 8, 256)  1024       ['conv4_block4_2_conv[0][0]']
ization)

conv4_block4_2_relu (Activatio  (None, 10, 8, 256)  0          ['conv4_block4_2_bn[0][0]']
n)

conv4_block4_3_conv (Conv2D)   (None, 10, 8, 1024)  263168    ['conv4_block4_2_relu[0][0]']

conv4_block4_3_bn (BatchNormal  (None, 10, 8, 1024)  4096      ['conv4_block4_3_conv[0][0]']
ization)

conv4_block4_add (Add)        (None, 10, 8, 1024)  0          ['conv4_block3_out[0][0]',
                                                               'conv4_block4_3_bn[0][0]']

conv4_block4_out (Activation)  (None, 10, 8, 1024)  0         ['conv4_block4_add[0][0]']

conv4_block5_1_conv (Conv2D)   (None, 10, 8, 256)   262400    ['conv4_block4_out[0][0]']

conv4_block5_1_bn (BatchNormal  (None, 10, 8, 256)  1024       ['conv4_block5_1_conv[0][0]']
ization)

conv4_block5_1_relu (Activatio  (None, 10, 8, 256)  0          ['conv4_block5_1_bn[0][0]']
n)

conv4_block5_2_conv (Conv2D)   (None, 10, 8, 256)   590080    ['conv4_block5_1_relu[0][0]']

conv4_block5_2_bn (BatchNormal  (None, 10, 8, 256)  1024       ['conv4_block5_2_conv[0][0]']
ization)

conv4_block5_2_relu (Activatio  (None, 10, 8, 256)  0          ['conv4_block5_2_bn[0][0]']
n)

conv4_block5_3_conv (Conv2D)   (None, 10, 8, 1024)  263168    ['conv4_block5_2_relu[0][0]']

conv4_block5_3_bn (BatchNormal  (None, 10, 8, 1024)  4096      ['conv4_block5_3_conv[0][0]']
ization)

conv4_block5_add (Add)        (None, 10, 8, 1024)  0          ['conv4_block4_out[0][0]',
                                                               'conv4_block5_3_bn[0][0]']

conv4_block5_out (Activation)  (None, 10, 8, 1024)  0         ['conv4_block5_add[0][0]']

conv4_block6_1_conv (Conv2D)   (None, 10, 8, 256)   262400    ['conv4_block5_out[0][0]']
```

conv4_block6_1_bn (BatchNormal (None, 10, 8, 256) 1024   ['conv4_block6_1_conv[0][0]']
ization)

conv4_block6_1_relu (Activatio (None, 10, 8, 256) 0   ['conv4_block6_1_bn[0][0]']
n)

conv4_block6_2_conv (Conv2D)   (None, 10, 8, 256)   590080   ['conv4_block6_1_relu[0][0]']

conv4_block6_2_bn (BatchNormal (None, 10, 8, 256) 1024   ['conv4_block6_2_conv[0][0]']
ization)

conv4_block6_2_relu (Activatio (None, 10, 8, 256) 0   ['conv4_block6_2_bn[0][0]']
n)

conv4_block6_3_conv (Conv2D)   (None, 10, 8, 1024) 263168   ['conv4_block6_2_relu[0][0]']

conv4_block6_3_bn (BatchNormal (None, 10, 8, 1024) 4096   ['conv4_block6_3_conv[0][0]']
ization)

conv4_block6_add (Add)   (None, 10, 8, 1024) 0   ['conv4_block5_out[0][0]',
                                                                'conv4_block6_3_bn[0][0]']

conv4_block6_out (Activation) (None, 10, 8, 1024) 0   ['conv4_block6_add[0][0]']

conv5_block1_1_conv (Conv2D)   (None, 5, 4, 512)   524800   ['conv4_block6_out[0][0]']

conv5_block1_1_bn (BatchNormal (None, 5, 4, 512)   2048   ['conv5_block1_1_conv[0][0]']
ization)

conv5_block1_1_relu (Activatio (None, 5, 4, 512)   0   ['conv5_block1_1_bn[0][0]']
n)

conv5_block1_2_conv (Conv2D)   (None, 5, 4, 512)   2359808   ['conv5_block1_1_relu[0][0]']

conv5_block1_2_bn (BatchNormal (None, 5, 4, 512)   2048   ['conv5_block1_2_conv[0][0]']
ization)

conv5_block1_2_relu (Activatio (None, 5, 4, 512)   0   ['conv5_block1_2_bn[0][0]']
n)

conv5_block1_0_conv (Conv2D)   (None, 5, 4, 2048)   2099200   ['conv4_block6_out[0][0]']

conv5_block1_3_conv (Conv2D)   (None, 5, 4, 2048)   1050624   ['conv5_block1_2_relu[0][0]']

conv5_block1_0_bn (BatchNormal (None, 5, 4, 2048) 8192   ['conv5_block1_0_conv[0][0]']
ization)

conv5_block1_3_bn (BatchNormal (None, 5, 4, 2048) 8192   ['conv5_block1_3_conv[0][0]']
ization)

conv5_block1_add (Add)   (None, 5, 4, 2048)   0   ['conv5_block1_0_bn[0][0]',
                                                                'conv5_block1_3_bn[0][0]']

conv5_block1_out (Activation) (None, 5, 4, 2048)   0   ['conv5_block1_add[0][0]']

conv5_block2_1_conv (Conv2D)   (None, 5, 4, 512)   1049088   ['conv5_block1_out[0][0]']

```
conv5_block2_1_bn (BatchNormal  (None, 5, 4, 512)   2048       ['conv5_block2_1_conv[0][0]']
ization)

conv5_block2_1_relu (Activatio  (None, 5, 4, 512)   0          ['conv5_block2_1_bn[0][0]']
n)

conv5_block2_2_conv (Conv2D)   (None, 5, 4, 512)    2359808    ['conv5_block2_1_relu[0][0]']

conv5_block2_2_bn (BatchNormal  (None, 5, 4, 512)   2048       ['conv5_block2_2_conv[0][0]']
ization)

conv5_block2_2_relu (Activatio  (None, 5, 4, 512)   0          ['conv5_block2_2_bn[0][0]']
n)

conv5_block2_3_conv (Conv2D)   (None, 5, 4, 2048)   1050624    ['conv5_block2_2_relu[0][0]']

conv5_block2_3_bn (BatchNormal  (None, 5, 4, 2048)  8192       ['conv5_block2_3_conv[0][0]']
ization)

conv5_block2_add (Add)         (None, 5, 4, 2048)   0          ['conv5_block1_out[0][0]',
                                                                'conv5_block2_3_bn[0][0]']

conv5_block2_out (Activation)  (None, 5, 4, 2048)   0          ['conv5_block2_add[0][0]']

conv5_block3_1_conv (Conv2D)   (None, 5, 4, 512)    1049088    ['conv5_block2_out[0][0]']

conv5_block3_1_bn (BatchNormal  (None, 5, 4, 512)   2048       ['conv5_block3_1_conv[0][0]']
ization)

conv5_block3_1_relu (Activatio  (None, 5, 4, 512)   0          ['conv5_block3_1_bn[0][0]']
n)

conv5_block3_2_conv (Conv2D)   (None, 5, 4, 512)    2359808    ['conv5_block3_1_relu[0][0]']

conv5_block3_2_bn (BatchNormal  (None, 5, 4, 512)   2048       ['conv5_block3_2_conv[0][0]']
ization)

conv5_block3_2_relu (Activatio  (None, 5, 4, 512)   0          ['conv5_block3_2_bn[0][0]']
n)

conv5_block3_3_conv (Conv2D)   (None, 5, 4, 2048)   1050624    ['conv5_block3_2_relu[0][0]']

conv5_block3_3_bn (BatchNormal  (None, 5, 4, 2048)  8192       ['conv5_block3_3_conv[0][0]']
ization)

conv5_block3_add (Add)         (None, 5, 4, 2048)   0          ['conv5_block2_out[0][0]',
                                                                'conv5_block3_3_bn[0][0]']

conv5_block3_out (Activation)  (None, 5, 4, 2048)   0          ['conv5_block3_add[0][0]']

flatten (Flatten)              (None, 40960)        0          ['conv5_block3_out[0][0]']

dense (Dense)                  (None, 128)          5243008    ['flatten[0][0]']

dense_1 (Dense)                (None, 7)            903        ['dense[0][0]']

==================================================================================================
Total params: 28,831,623
```

Trainable params: 5,243,911
Non-trainable params: 23,587,712
_____