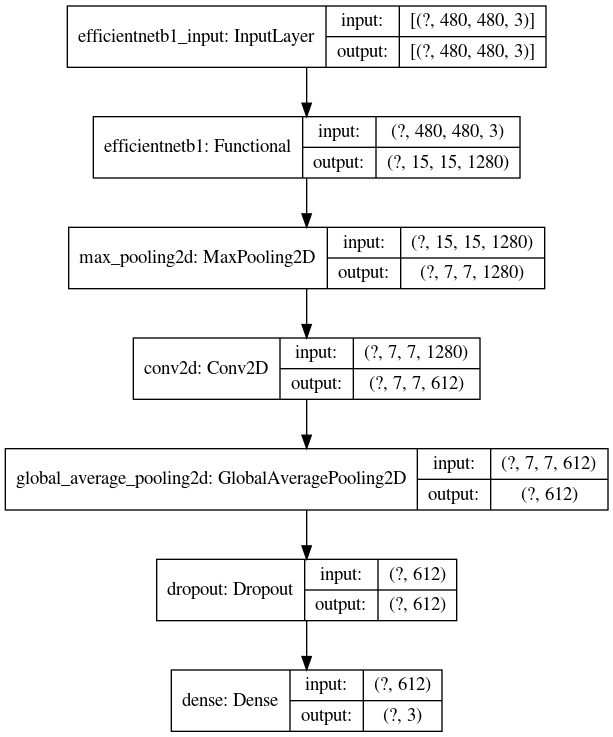
**Description of Best Model**

EfficientNetB1

We have decided to use EfficientNet which guarantees accuracy while ensuring efficiency under the adoption of compound scaling method [1]. We made use of transfer learning by using EfficientNetB1 as the base model with weights pre-trained on imagenet so the model can converge faster. On top of the base model, we added a max pooling layer, followed by another convolutional layer with (1,1) sized kernels number of filters for the purpose of dimensionality reduction and layer regularization with a L2 norm. We then aggregate the information of each feature map using a global average pooling layer and feed the output to a dropout layer to prevent overfitting. Lastly, the softmax layer will calculate the probability of each image belonging to each class respectively and output the class with maximum probability. With this model, we could have achieved the highest private leaderboard score of 0.97945, but unfortunately we overly relied on the result of the public leaderboard and therefore overfitted our model by increasing the number of filters of the convolutional layer and selected a submission with a much lower score.

**Experimental Study**

Model Selection

To decide on the type of EfficientNet to be used, we have performed cross validation over all possible EfficientNet versions and found out that B1 has the best performance. We did a 4-fold cross validation using EfficientNets B0 to B7 as the base model with a cascade of a global average pooling layer and a dense layer of 3 units as the classifier. Each model is trained for 10 epochs per fold with the same hyperparameters/configurations and evaluated across all folds on average.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Loss | Accuracy | Validation Loss | Validation Accuracy |
| B0 | 0.21728 | 0.91139 | 0.25486 | 0.88464 |
| B1 | 0.20817 | 0.91718 | 0.24826 | 0.89468 |
| B2 | 0.23350 | 0.90512 | 0.27976 | 0.87766 |
| B3 | 0.24214 | 0.90072 | 0.28033 | 0.88317 |
| B4 | 0.23236 | 0.90721 | 0.26098 | 0.88840 |
| B5 | 0.21230 | 0.91463 | 0.26958 | 0.88685 |
| B6 | 0.24392 | 0.90409 | 0.29367 | 0.87939 |
| B7 | 0.23523 | 0.90584 | 0.28078 | 0.88591 |

Data Augmentation

We have performed data augmentation by flipping the data vertically and horizontally in order to let the model learn to identify the Lesion area in multiple manners. When we observed the dataset, we found that the brightness of the images varies so we randomly adjusted the brightness of the images to increase the model’s ability to recognize the features regardless of brightness. At first, we thought it would be a good idea to add contrast, random cropping and scaling to the images because we spotted that not all images have the same scale and some features are not obvious. Contrasting can help to differentiate the features from the background and cropping/scaling may be useful to centre each image, however all of them seemed to harm the performance after validation.

Hyperparameter Tuning

We used the keras hyperparameter tuner to find the best hyperparameters combination using the Hyperband strategy, which optimizes the speed of random search through adaptive resource allocation and early-stopping. [2] We tuned the number of filters in the convolutional layer, dropout rate and tried out multiple optimizers with various learning rates to find the best combination. The top five result is summarized in the following table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Optimizer | Learning Rate | Number of Filters | Dropout Rate | Validation Accuracy |
| Adam | 0.001 | 400 | 0.5 | 1.0 |
| RMSProp | 0.0009 | 1000 | 0.3 | 1.0 |
| SGD | 0.0004 | 1000 | 0.5 | 1.0 |
| SGD | 0.0026 | 400 | 0.3 | 0.98 |
| Adam | 0.001 | 1000 | 0.4 | 0.96 |

We then used the best tuned set of hyperparameters as the initial starting value. For better performance, we have used early stopping to prevent overfitting. We also used the keras callback for reducing learning rate at plateau to maintain an adaptive learning rate for better convergence. Model checkpoint is also used to save the best model with highest validation accuracy.

Reference:

[1] Tan, M. and Le, Q., 2020. *Efficientnet: Rethinking Model Scaling For Convolutional Neural Networks*. [online] arXiv.org. Available at: <https://arxiv.org/abs/1905.11946> [Accessed 5 November 2020].

[2] Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A. and Talwalkar, A., 2020. *Hyperband: A Novel Bandit-Based Approach To Hyperparameter Optimization*. [online] arXiv.org. Available at: <https://arxiv.org/abs/1603.06560> [Accessed 5 November 2020].