# 1 INTRODUCTION

Two methods are employed to tune the baseline model: Random Search with Keras Tuner and Transfer Learning with the pre-trained model VGG16 [1]. The random search gives the best combination of hyperparameters which are used to train on the training set (see in Table 1) and in the transfer learning, a model combines the convolution and max poolings layers of the VGG16 model and self-defined dense layers are trained (see in Table 2). The project utilizes various techniques to control overfitting and also explores different hyperparameters to improve the model performance, which will be explained in the following sections. In this report, the $1^{st}$, $2^{nd}$, and $3^{rd}$ model respectively represents the baseline model, the best model from the random search, and the model used the transfer learning.

# 2 OVERFITTING CONTROL

To control the overfitting, early stopping, model checkpoint,$L_1$ and $L_2$ regularizer, and dropout layers are applied in the tuning stage. The use of early stopping enables the model stop training after 5 continuous epochs (patience = 5) where the loss on the validation set does not drop below the lowest-ever loss. Early stopping, cooperated with the function *model_checkpoint* in Keras, can retain the weights of the model with lowest validation loss, namely the last $6^{\text{th}}$ model. In both the $2^{nd}$ and the $3^{rd}$ models, $L_1$ and $L_2$ regularizer and dropout layers are included in the training (see in 3.1.2 and 3.2).

# 3 HYPERPARAMETER TUNING

## 3.1 *Random Search*

The RandomSearch() in *keras_tuner* library enables us to try a wide range of hyperparameters and find the best possible configuration towards an objective - in our case, the maximum of the validation accuracy. Our random search is based on the baseline model in two senses. Firstly, the architecture of the baseline model is mostly duplicated, except that one more convolution layer is added before the last max pooling layer. Secondly, most hyperparameters in the baseline model, such as the number of filters, filter size, the number of fully connected (FC) layers, and the number of neurons in each FC layer, are contained in the range of the hyperparameter

[1] Three models were developed in the code while this report will analyze the two models from section 9 to 14 in the code since the model in section 5 is a preparation of the random search.

settings. The effect of different optimization and regularization are also compared in the network.

### 3.1.1 *Convolution Layer and Fully Connected Layer*

All five convolution layers have the same minimum number of filters (In the code, it is mistakenly named "filter size"), 64, while their maximum increases as the network go deeper. The kernel size is able to change from (3,3) to (7,7). Each convolution layer is with zero-padding and can opt for a stride size between 1 and 2. The number of FC layers in the baseline model is preserved in the random search. However, in two FC layers, the neuron size of the second one is set to have a minimum of 0 to trial whether this layer is necessary. Both layers are assigned with a Boolean option of the switch-on or -off of the bias.

### 3.1.2 *Regularization and Optimization*

Three methods are used for regularization: $L_2$ regularizer, dropout layer, and early stopping. The first two are defined in the model creation stage. To reduce the time of the random search, each of the convolution and FC layers has two options for the $\lambda$ of $L_2$ regularizer, 0 or 0.01. A dropout layer follows the first FC layer with the dropout rate ranging from 0 to 0.5 with a step size of 0.05, in order to balance the overfitting and underfitting. The early stopping application was described in Section 2. In the model compile step, the optimizer Adam was assigned 3 possible learning rates, 0.01, 0.001, 0.0001.

## 3.2 *Transfer Learning*

Two FC layers are added between the last max pooling layer of the VGG16 model and the output layer in the transfer learning, where the first is with the regularization of $L_1$ and $L_2$ on default $\lambda$ ($L_1 = 0.1$, $L_2 = 0.1$). A dropout layer that drops 50% of the neurons is followed by the first FC layer to combat the overfitting. In model fitting, besides the same early stopping and model checkpoint settings, the callback function $ReduceLROnPlateau()$ that decreases the learning rate when the loss does not lessen for 5 epochs (patience = 5) is in use to help improve the model performance.

# 4 RESULTS

## 4.1 *Metrics*

The metrics (see in Table 3) indicate that the $3^{rd}$ model has the best performance, and both of the two tuned models handle the overfitting problem properly. The effects of overfitting control can also be observed from the loss and accuracy in each epoch during model training (See in Figure 1). Since the objective of the early stopping for the two tuned models is to minimize the loss on the validation set, the effect of the early stopping is more observable in the loss on both training and validation sets. The plots show that the $2^{nd}$ model's loss in the training and validation set during the model fit has the closest curves compared to the other two models.

## 4.2 *Confusion Matrices*

Both count-based and normalized confusion matrices (CMs) are reported. The diagonal cells of the normalized CMs are the sensitivity of each class. From the CMs of the baseline model on both validation and test sets (see in Figure 2), it can be noticed that the sensitivity of classes "glacier" and "sea" are less than or equal to 0.7, and the sensitivity of the class "mountain" drops from 0.83 to 0.79, showing that the sensitivity per class are imbalanced and the model has relatively high variance. The sensitivity per class from the two tuned models is more balanced and less variant. Also, the prediction difficulty for 6 classes is changed. In the $2^{nd}$ model, "glacier" and "sea" are still the hardest to predict, while in the $3^{rd}$ model, "glacier" and "mountain" have the lowest sensitivity.

## 4.3 *AUC-ROC Curves*

All the AUC-ROC curves are close to the left-top (see Figure 5), denoting a good balance between sensitivity and specificity.

# 5 DISCUSSION

As mentioned in Section 4, the $2^{nd}$ and the $3^{rd}$ models improve the performance from the baseline model on three aspects: (1) The overall performance measured by macro accuracy, sensitivity, specificity, F1-score is improved to some extent; (2) the predictive ability of individual classes tends to be more balanced and less variant; (3) the overfitting observed in the baseline model is well-controlled.

## 5.1 *Neural Network Architecture*

The architecture of the $2^{nd}$ model (see in Table 4), shares some similar features with the $3^{rd}$ model: the number of filters increases as the convolution layer goes deeper, while the FC layer that is closer to the output layer has fewer neurons than the deeper layers. According to the study conducted by Wouters, Arribas, Gierlichs, and Preneel (2020), adding a convolution layer and amplifying the filter size could improve the model towards better performance. Besides, both the $2^{nd}$ and the $3^{rd}$ model are equipped with wider FC layers than the baseline model. From the study by Basha, Dubey, Pulabaigari, and Mukherjee (2020), there is a balance between the depth and the width of a CNN architecture. Shallow architecture requires wide FC layers and verse vice. Among our models, though the $2^{nd}$ model has fewer convolution layers and fewer neurons in the FC layers compared to the $3^{rd}$ model, it has a greater filter size and one more FC layer, which might explain why those two models have close performance measured by metrics in Section 3.

## 5.2 *Regularization*

The employment of the regularization, $L_1$, $L_2$, and dropout layer, has been proved to have effects on controlling overfitting (Jayech, 2017). It is also observed from the results of the two tuned models as shown in Figure 1 and illustrated in Section 4. Differing from the baseline model, the FC layers are penalized with $L_1$ or $L_2$ regularizers in the two tuned models. Also, in both the $2^{nd}$ and the $3^{rd}$ models, a single dropout layer with a high dropout rate (0.45 and 0.50 respectively) is inserted following the first FC layer that contains the greatest number of neurons. The design of the layer position and the optimistic dropout rate from the random search is supported by the study by Park and Kwak (2017) that the regularization is stronger with deep FC layers than shallow ones, and a deeper layer is tended to be more effective with a higher dropout rate.

## 5.3 *Others*

Future research could investigate the other factors that are not included in the previous discussion but might also explain part of the improvement from the baseline to the tuned models, such as the change of the learning rate Adam, the choices of the optimizer, the padding type, and the stride size. However, those inferences are not sufficiently backed by the setting of the three models in this project.

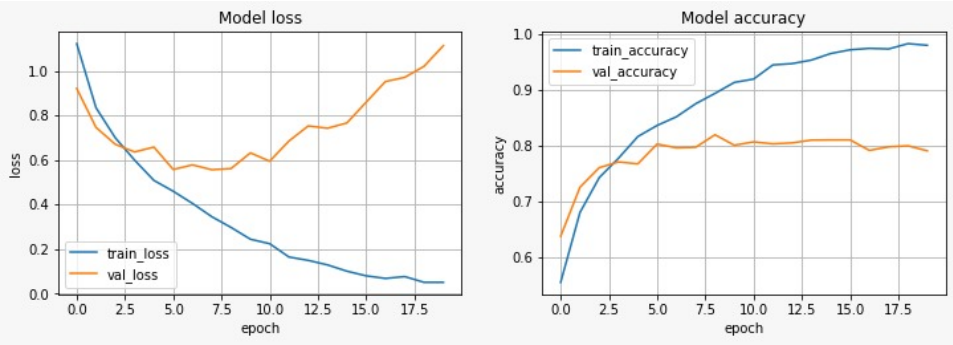
## REFERENCES

Basha, S. H., Dubey, S. R., Pulabaigari, V., & Mukherjee, S. (2020, 2). Impact of fully connected layers on performance of convolutional neural networks for image classification. *Neurocomputing*, *378*, 112-119. doi: 10.1016/J.NEUCOM.2019.10.008

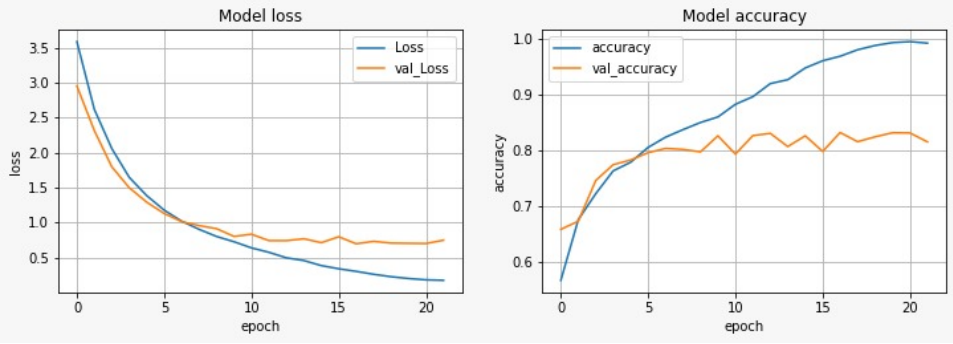
Jayech, K. (2017). Regularized deep convolutional neural networks for feature extraction and classification. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *10635 LNCS*, 431-439. Retrieved from https://link.springer.com/chapter/10.1007/978-3-319-70096-0_45 doi: 10.1007/978-3-319-70096-0_45/COVER

Park, S., & Kwak, N. (2017). Analysis on the dropout effect in convolutional neural networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *10112 LNCS*, 189-204. Retrieved from https://link.springer.com/chapter/10.1007/978-3-319-54184-6_12 doi: 10.1007/978-3-319-54184-6_12/COVER

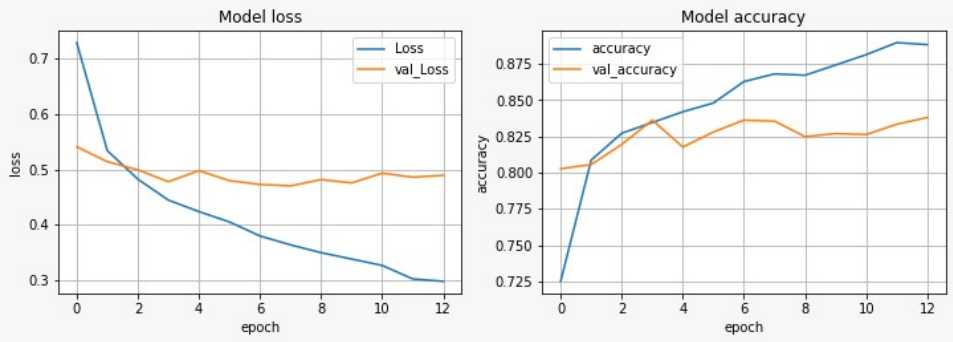
Wouters, L., Arribas, V., Gierlichs, B., & Preneel, B. (2020, 6). Revisiting a methodology for efficient cnn architectures in profiling attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, *2020*, 147-168. Retrieved from https://tches.iacr.org/index.php/TCHES/article/view/8586 doi: 10.13154/TCHES.V2020.I3.147-168




(a) Baseline

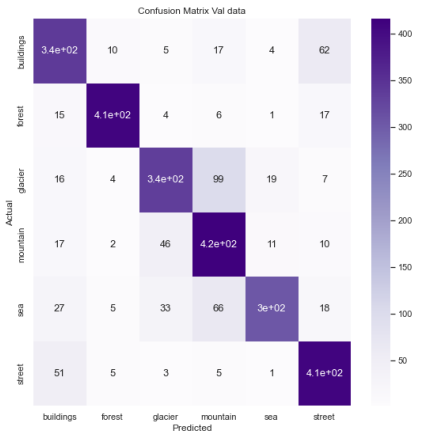



(b) Random search

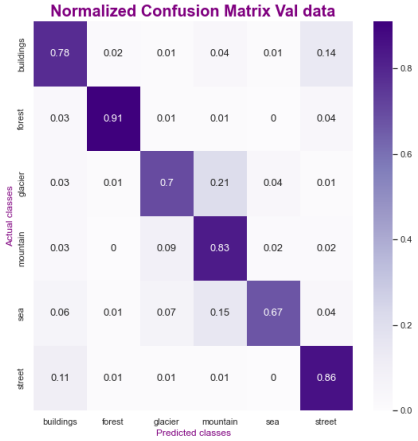



(c) Transfer learning

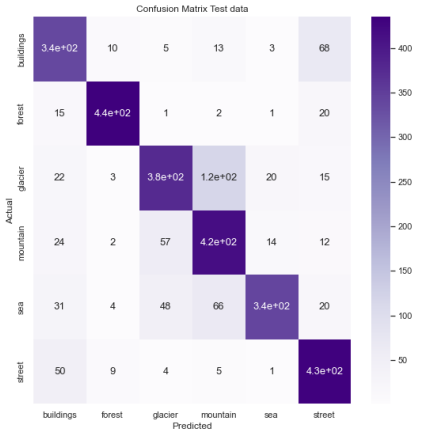
Figure 1: The training and validation losses and accuracies on the training and validation set for three models



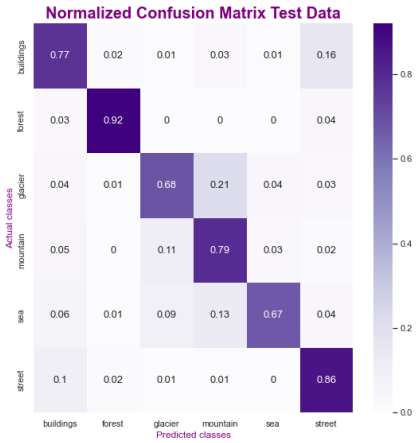

(a) Count-based CM on validation set




(b) Normalized CM on validation set

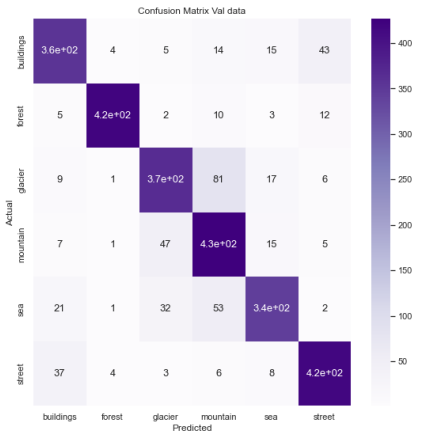



(c) Count-based CM on test set

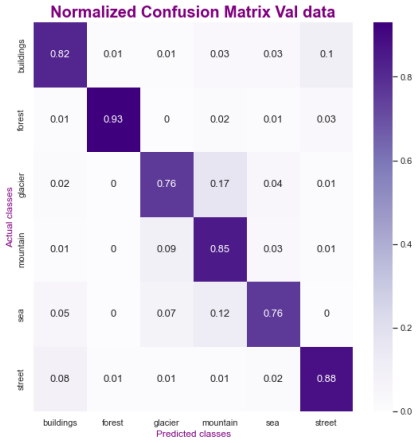



(d) Normalized CM on test set

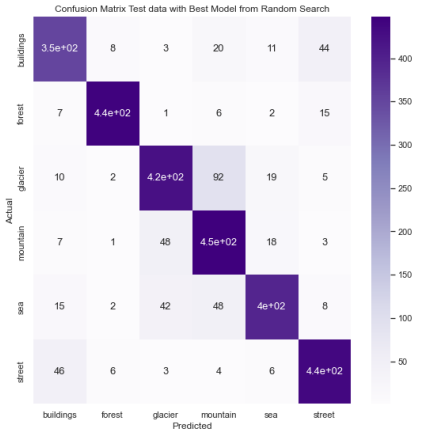
Figure 2: Confusion matrices of baseline model



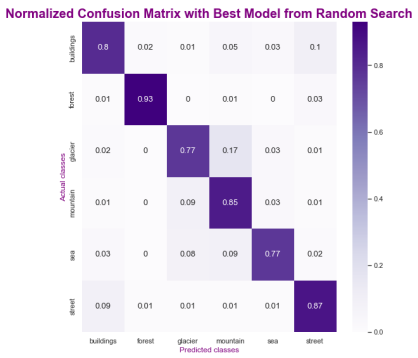

(a) Count-based CM on validation set




(b) Normalized CM on validation set

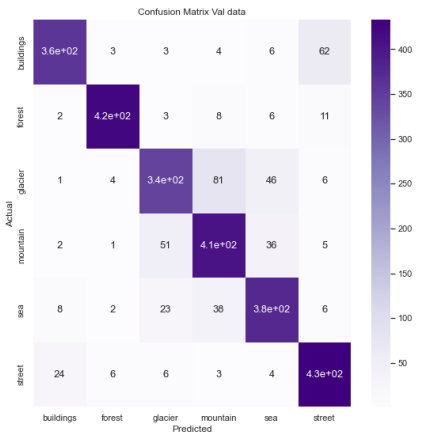



(c) Count-based CM on test set

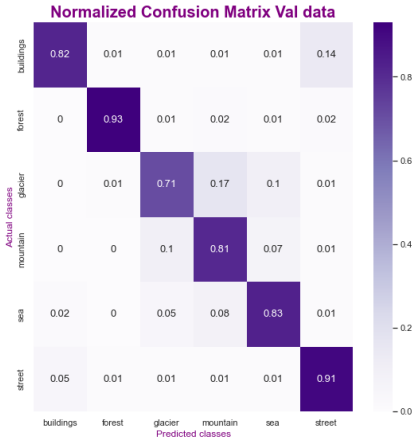



(d) Normalized CM on test set

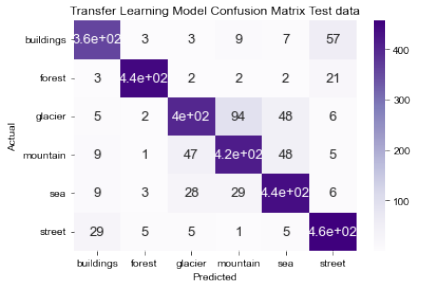
Figure 3: Confusion matrices of the best model from random search



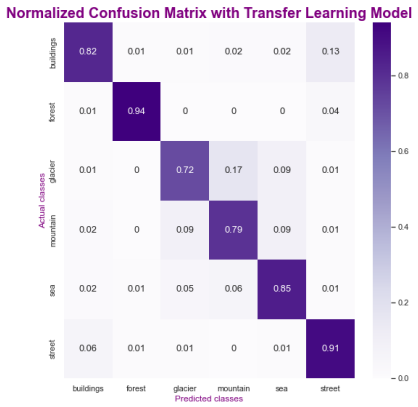

(a) Count-based CM on validation set




(b) Normalized CM on validation set

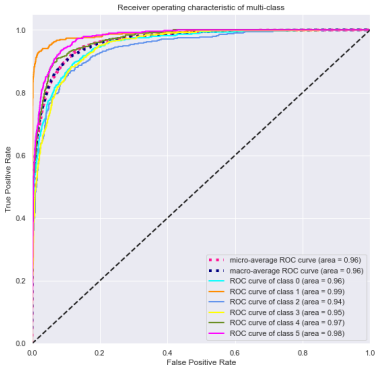



(c) Count-based CM on validation set

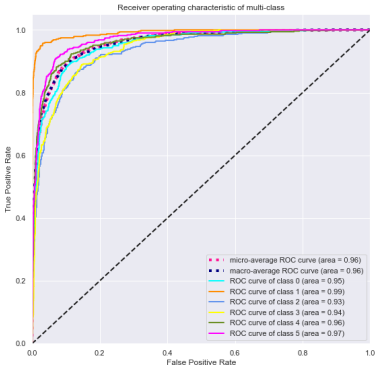



(d) Normalized CM on validation set

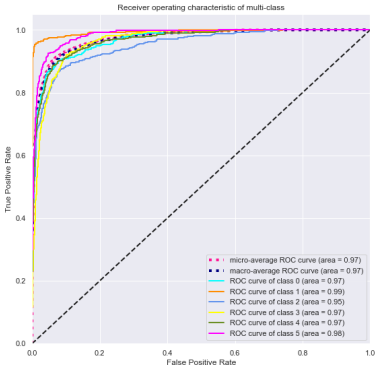
Figure 4: Confusion matrices of the model with transfer learning



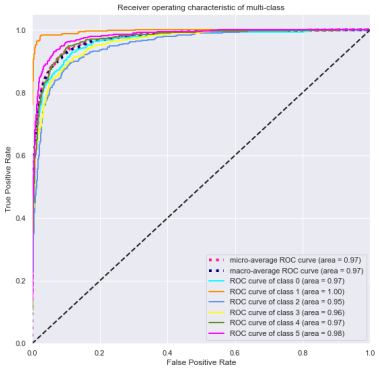

(a) Baseline, validation set



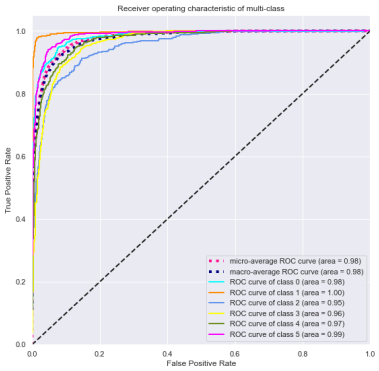

(b) Baseline, test set



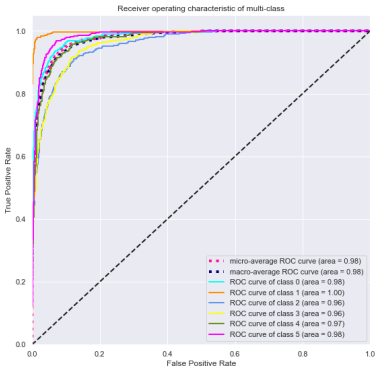

(c) Random search, validation set




(d) Random search, test set




(e) Transfer learning, validation set




(f) Transfer learning, test set

Figure 5: AUC-ROC curve

Table 1: The model summary of the best model from random search

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d (Conv2D) | (None, 64, 64, 128) | 3584 |
| max_pooling2d (MaxPooling2D) | (None, 32, 32, 128) | 0 |
| conv2d_1 (Conv2D) | (None, 32, 32, 256) | 295168 |
| max_pooling2d_1 (MaxPooling2D) | (None, 16, 16, 256) | 0 |
| conv2d_2 (Conv2D) | (None, 8, 8, 192) | 2408640 |
| conv2d_3 (Conv2D) | (None, 4, 4, 384) | 3613056 |
| max_pooling2d_2 (MaxPooling2D) | (None, 2, 2, 384) | 0 |
| flatten (Flatten) | (None, 1536) | 0 |
| dense (Dense) | (None, 160) | 245920 |
| dropout (Dropout) | (None, 160) | 0 |
| dense_1 (Dense) | (None, 32) | 5152 |
| dense_2 (Dense) | (None, 6) | 198 |
| Total params: 6,571,718 | | |
| Trainable params: 6,571,718 | | |
| Non-trainable params: 0 | | |

Table 2: The model summary of tuned model with transfer learning

| Layer (type) | Output Shape | Param # |
|---|---|---|
| vgg16 (Functional) | (None, 2, 2, 512) | 14714688 |
| flatten (Flatten) | (None, 2048) | 0 |
| dense (Dense) | (None, 256) | 524544 |
| dropout (Dropout) | (None, 256) | 0 |
| dense_1 (Dense) | (None, 6) | 1542 |
| Total params: 15,240,774 | | |
| Trainable params: 526,086 | | |
| Non-trainable params: 14,714,688 | | |

Table 3: Performance for three models

| | Baseline | | Random Search | | Transfer Learning | |
|---|---|---|---|---|---|---|
| | Val | Test | Val | Test | Val | Test |
| Loss | 1.1127 | 1.2112 | 0.6935 | 0.7053 | 0.4705 | 0.4560 |
| Accuracy | 0.7910 | 0.7793 | 0.8327 | 0.8327 | 0.8356 | 0.8353 |
| Sensitivity | 0.7905 | 0.7824 | 0.8326 | 0.8340 | 0.8365 | 0.8382 |
| Specificity | 0.9580 | 0.9558 | 0.9664 | 0.9664 | 0.9670 | 0.9670 |
| F1-Score | 0.7911 | 0.7795 | 0.8334 | 0.8333 | 0.8355 | 0.8350 |

Table 4: The hyperparameters of the best model from random search

| | Kernel Size | Number of Filters | Stride | Number of Neurons | L2 | Dropout Rate | With Bias? |
|---|---|---|---|---|---|---|---|
| Conv2D | (3,3) | 128 | 1 | | 0,01 | | |
| MaxPooling2D | | | | | | | |
| Conv2D | (3,3) | 256 | 1 | | 0 | | |
| MaxPooling2D | | | | | | | |
| Conv2D | (7,7) | 192 | 2 | | 0 | | |
| Conv2D | (7,7) | 384 | 2 | | 0 | | |
| MaxPooling2D | | | | | | | |
| Dense | | | | 160 | 0.01 | | True |
| Dropout | | | | | | | |
| Dense | | | | 32 | 0 | 0.45 | True |
| Dense (Output) | | | | 6 | | | |