Computer Vision for Remote Sensing - Coding Challenge

Jakob Hackstein

Abstract. This summary presents three deep learning models which classify images from the remote sensing dataset as part of the coding challenge. The initial baseline achieves an accuracy of 20.43% as it suffers from an imbalanced dataset and small CNN architecture. After dataset enhancements, a refined baseline model increases the accuracy to 47.31%. Finally, a transfer learning model containing pretrained-weights from EfficientNet with an accuracy of 83.15% is presented.

Link to source code on GitHub: https://github.com/jakhac/rs-classification

1. Introduction. As part of the project *Computer Vision for Remote Sensing* a coding challenge was introduced. The goal of this challenge is to develop deep learning models which classify 256×256-pixel images into earth features like agriculture, forest and beach. The documentation and summary of results is presented here and structured as follows: First, the dataset is analyzed and potential problems are discussed. Then, the approach regarding model training and evaluation is explained. After a detailed presentation of three models, the main results are summarized and further ideas for improvement are mentioned.

2. Data Analysis. Let us now analyze the data. The dataset contains $1,370$ images distributed over 21 classes. In the domain of deep learning, this is a rather small dataset. As a consequence, overfitting might pose a problem since few samples per class only provide a limited amount of variance and number of features to learn. Furthermore, the distribution of classes is imbalanced, as visible in Figure 1. In particular, the most represented class *agriculture* (85 samples) contains more than three times as many images as the least represented class *sparseresidential* (27 samples). Training with an imbalanced dataset decreases a model's performance since more populated classes are more likely to influence the parameters during training.

3. Workflow. Let us now discuss the approach to train and evaluate models. The workflow essentially consists of the following fundamental steps. First, the training and test images are preprocessed and subsequently stored in two corresponding arrays. The preprocessing functions ensure that each pixel value is in the interval $[0, 255]$ and the correct target size is chosen. Furthermore, the training set is split into training and validation sets using a $80/20$ ratio. The resulting number of images per set is depicted in Table 1. Second, a deep learning model is created and trained using the keras library. In this coding challenge, all models are compiled with an Adam-optimizer and the categorical crossentropy loss function to ensure comparability. Third, the progress of loss and accuracy along all epochs of both training and validation data is plotted. This step aims to identify potential signs of over- or underfitting. Finally, several candidate models from different epochs predict classes for the test set. This step is done manually since the number of epochs is low and only few epochs manage to improve the model. The best model is then chosen according to three metrics: accuracy, precision and recall. Accuracy is computed as a standard measure to evaluate prediction success of a model. However,

Set	Images
Training	872
Validation	279
Test	219

Table 1. Images per set.

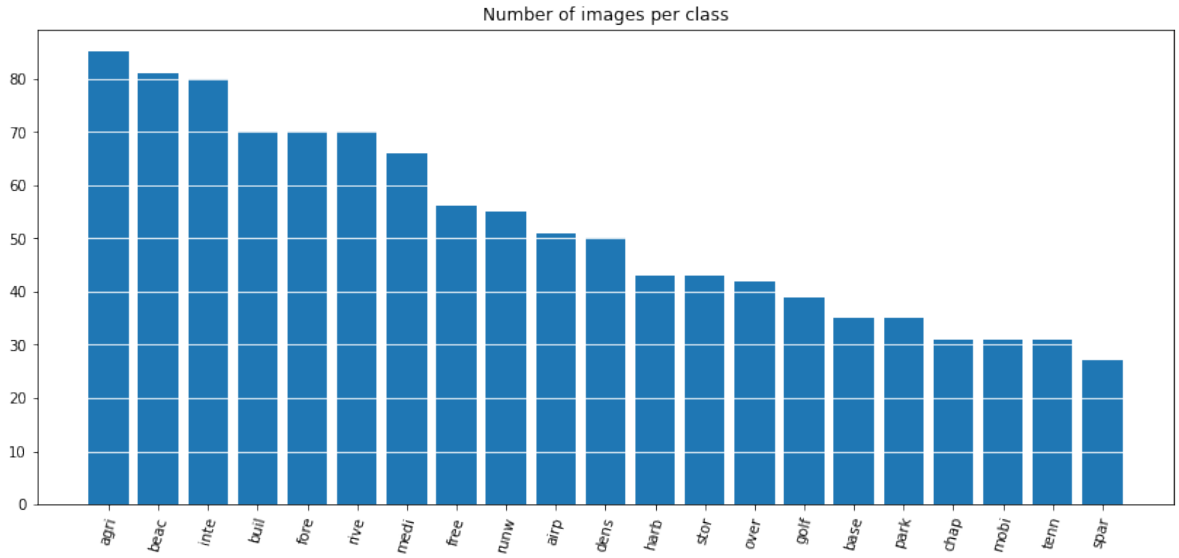


Figure 1. Original distribution of classes of training images.

this score might be misleading due to the dataset imbalance as high prediction success does not necessarily imply prediction capacity for smaller classes. Therefore, precision and recall scores are computed to indicate reliable measurements, even if the dataset is imbalanced.

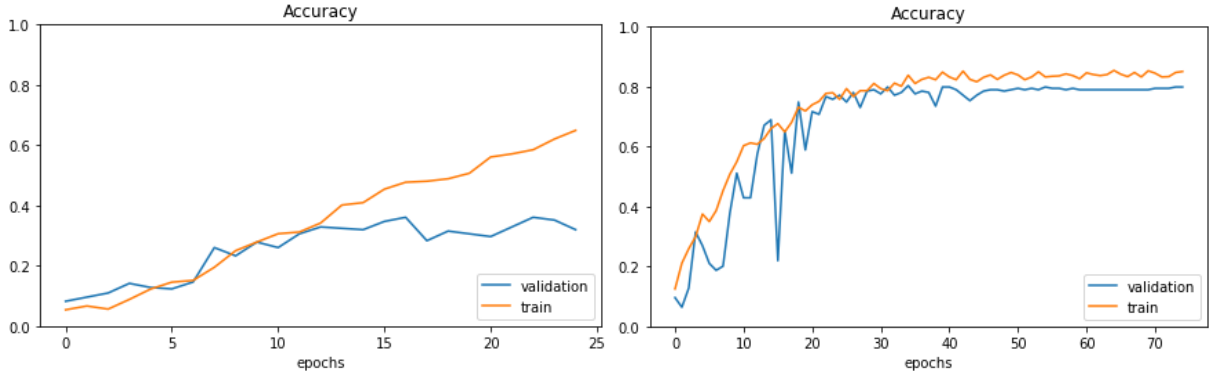
In several iterations of this workflow, flaws in model design and data enhancement were identified during the evaluation step and subsequently refined in the succeeding iteration. Thus, incrementally improving models were developed.

4. Models. Let us now take a closer look at the baseline model and two additional models with significant progress to its preceding model. Since the evaluation is closely related to the design choices of the succeeding model, model presentation and evaluation are combined in this section.

4.1. BASELINE (V1). The baseline model, also BASELINE_V1, essentially follows the workflow as outlined in Section 3 without any additional steps. Note that this model was trained on the initial dataset without any modifications regarding imbalance. The BASELINE_V1 model consists of a straightforward CNN architecture with four convolutions, all having 32 filters, and a dense layer, before the classes are predicted in the final layer. As visible in the Figure 2a, the baseline model seems to overfit after estimated 13 epochs since the training and validation accuracy diverge. To be precise, the model learns features which are specific to the training set and therefore fails to apply them in the validation set.

After 17 epochs, the model scores best when predicting the test set. In numbers, the BASELINE_V1 model achieves an accuracy of 20% and rather low scores for precision (22%) and recall (20%), respectively. When analyzing the confusion matrix in Figure 3 and the distribution of predicted classes in Figure 6, it becomes apparent that this performance is due to the imbalanced dataset as discussed in Section 2. In particular, the most represented class *agriculture* in the training set is also the most predicted class by the model. Therefore, the first column corresponding to the *agriculture* class in the confusion matrix is highly populated whereas the diagonal is rather empty.

Based on the evidence of the evaluation, the most important adjustments to improve performance are balancing the dataset, providing more images per class and tuning hyperparameters in the CNN. Moreover, the CNN can be enlarged, given that more samples are in each class.



(a) Train- and validation accuracy of BASELINE_V1

(b) Train- and validation accuracy of BASELINE_V2

Figure 2. Train- and validation accuracy.

4.2. BASELINE (V2). After applying the previously proposed ideas in several workflow iterations, the BASELINE_V2 model is able to outperform its predecessor. The most notable changes include balancing the dataset and data augmentation. The former is done by passing a list of weights to the training function such that underrepresented classes have more impact than overrepresented classes. The latter generates artificial images based on slight changes to real images of the training set. According to test runs, minor changes to height, width and rotation as well as horizontal mirroring yield the best results. Since the dataset is now balanced and contains more variation, a larger CNN model with more filters in deeper convolution layers is used. In numbers, there are five convolution layers where the number of filters increases from 32 to 256. In addition, batch normalization layers and increasing dropout percentages are implemented in order to prevent overfitting. Furthermore, during training the learning rate is now reduced by a factor of 0.5 whenever three consecutive epochs did not improve the validation accuracy.

During evaluation of the BASELINE_V2 model on the test set, the version trained for 35 epochs performs best. This particular model achieves an accuracy of 47% while the precision and recall scores also increase to 44% and 47%, respectively. The model does not overfit anymore since the training and validation accuracy converges, as visible in Figure 2b. Classes are now predicted more uniformly (Figure 7) and the diagonal along the confusion matrix (Figure 4) is visible.

However, accuracy during validation is notably higher than accuracy on the test set. In an attempt to close this gap and increase accuracy even further, a third model based on transfer learning techniques is discussed in the following.

4.3. TRANSFER LEARNING MODEL. Although data augmentation is able to enhance the quality of training data and therefore prediction success, it cannot compensate a large dataset. Therefore, the final model is created using transfer learning techniques. In particular, the feature extraction of the EfficientNetB0 network trained on a large image database is combined with a new classification layer. The choice for EfficientNetB0 is due to its performance/size ratio. During

training, the weights of the pretrained base model are frozen whereas the top layer is trained to classify the extracted features according to our classes. The automatically decreasing learning rate as explained in BASELINE_V2 is applied in this model, too.

The transfer learning model classifies 83% of test images correctly. Furthermore, it achieves a precision of 86% and a recall of 83%. These improvements are also visible in the confusion matrix depicted in Figure 5 and in a nearly uniform distribution of predicted classes in Figure 8.

5. Conclusion and Outlook. In conclusion, decisive improvements occur after balancing the dataset and adjusting the CNN architecture. Furthermore, adding the feature extraction of a pretrained model nearly doubles all metrics again. The results of all models are summarized again in Table 2.

	Accuracy	Precision	Recall
baseline v1	20.43	22.33	20.43
baseline v2	47.31	44.91	47.31
transfer learning	83.15	86.49	83.15

Table 2. All metrics for all models in %

Although all metrics vastly improved from the initial BASELINE_V1 model during this research, additional steps might improve the classification even more. For instance, the lack of overall training samples could be resolved by generating artificial images using state of the art GANs since this approach allows feature extraction to learn more specific details for each class. Another way to cope with the imbalance of the dataset could be the use of the focal loss function. Further, tuning the hyperparameters is a promising but time consuming step to take. Regarding the transfer model, a common technique to slightly increase performance is to unfreeze all layers and train the CNN end-to-end with a small learning rate.

6. Appendix.

6.1. Confusion Matrices.

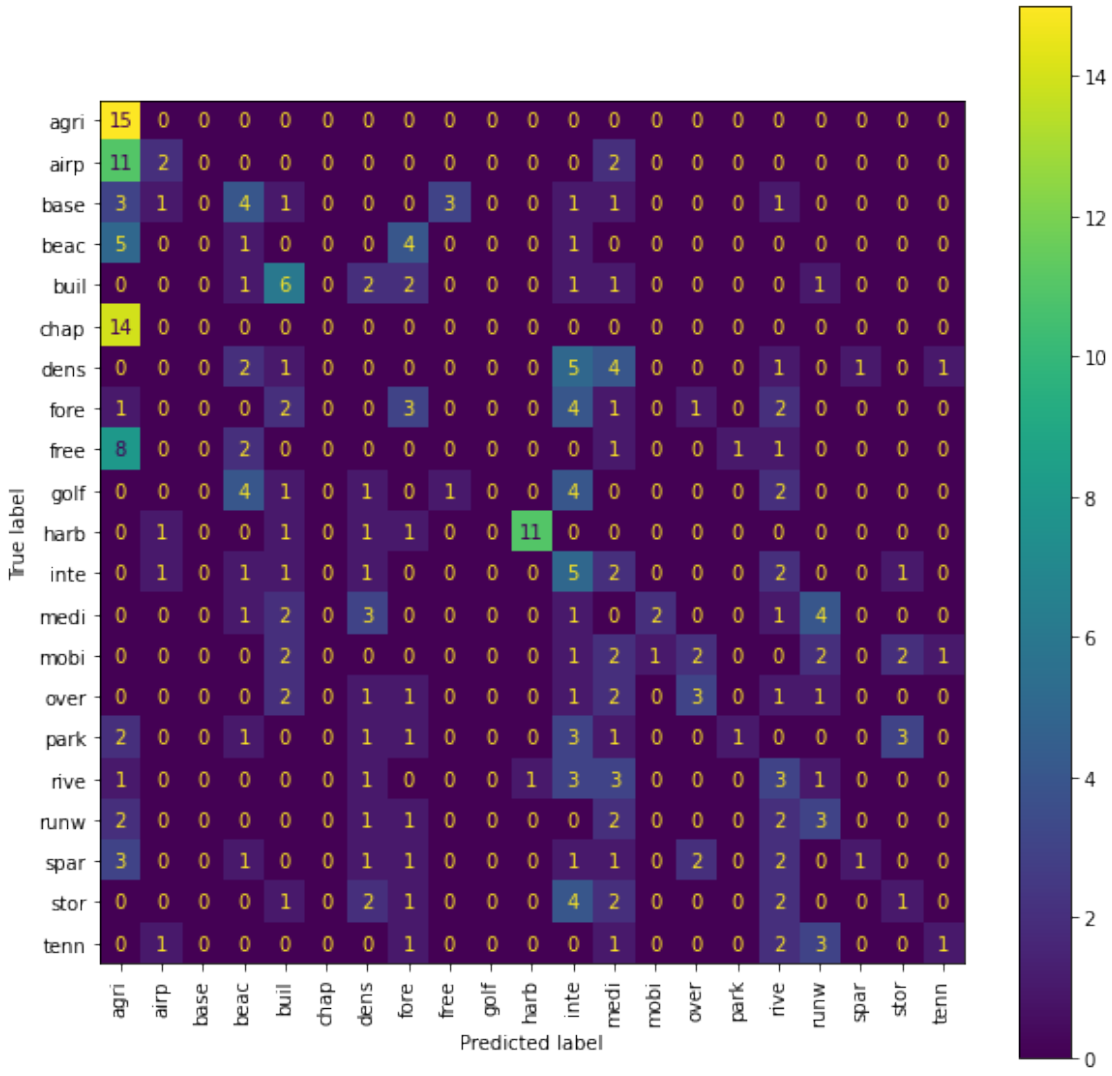


Figure 3. Confusion matrix for BASELINE_V1. Note that numbers are not normalized for better readability. The average test class contains 13.29 samples.

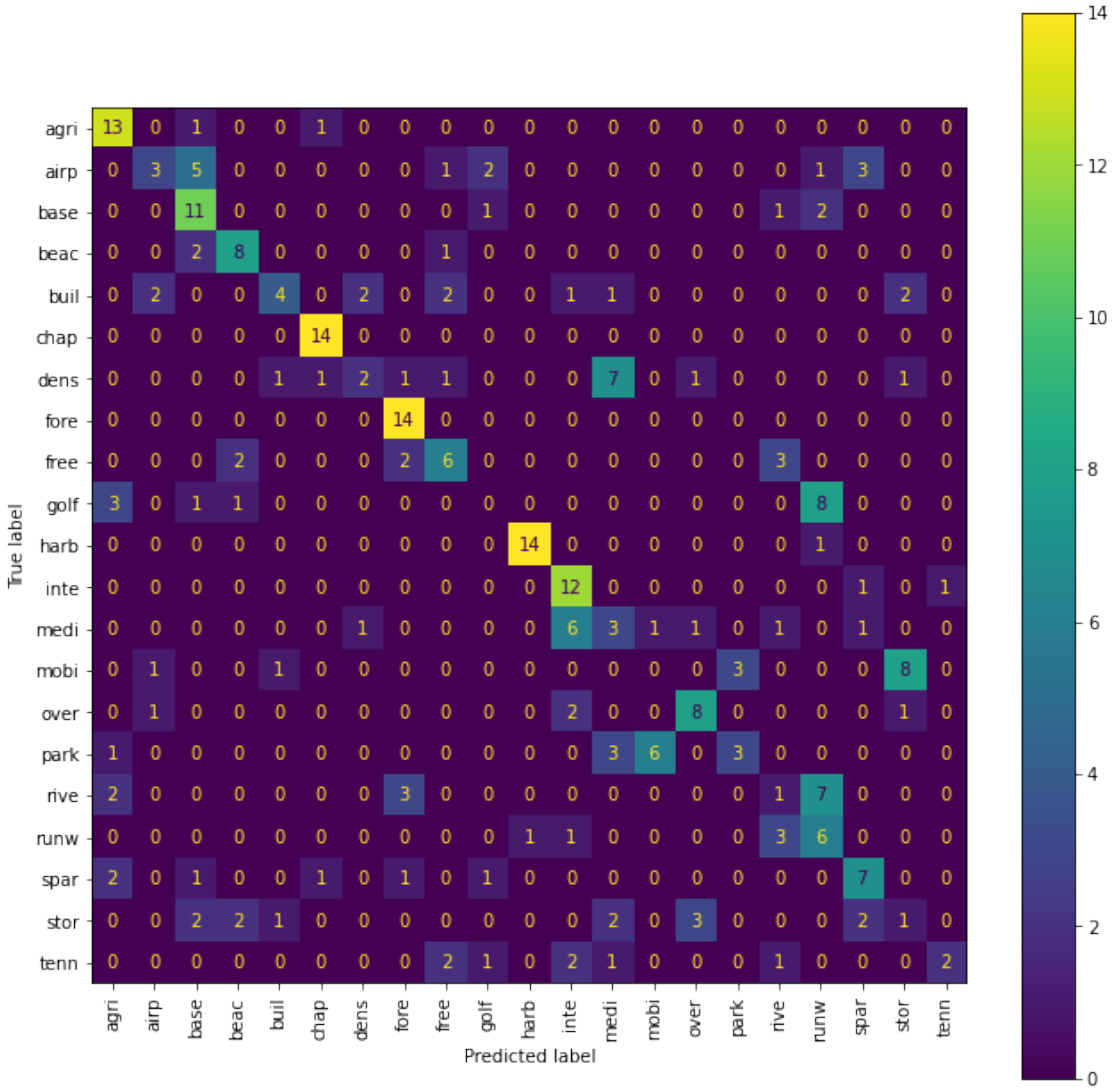


Figure 4. Confusion matrix for BASELINE_V2. Note that numbers are not normalized for better readability. The average test class contains 13.29 samples.

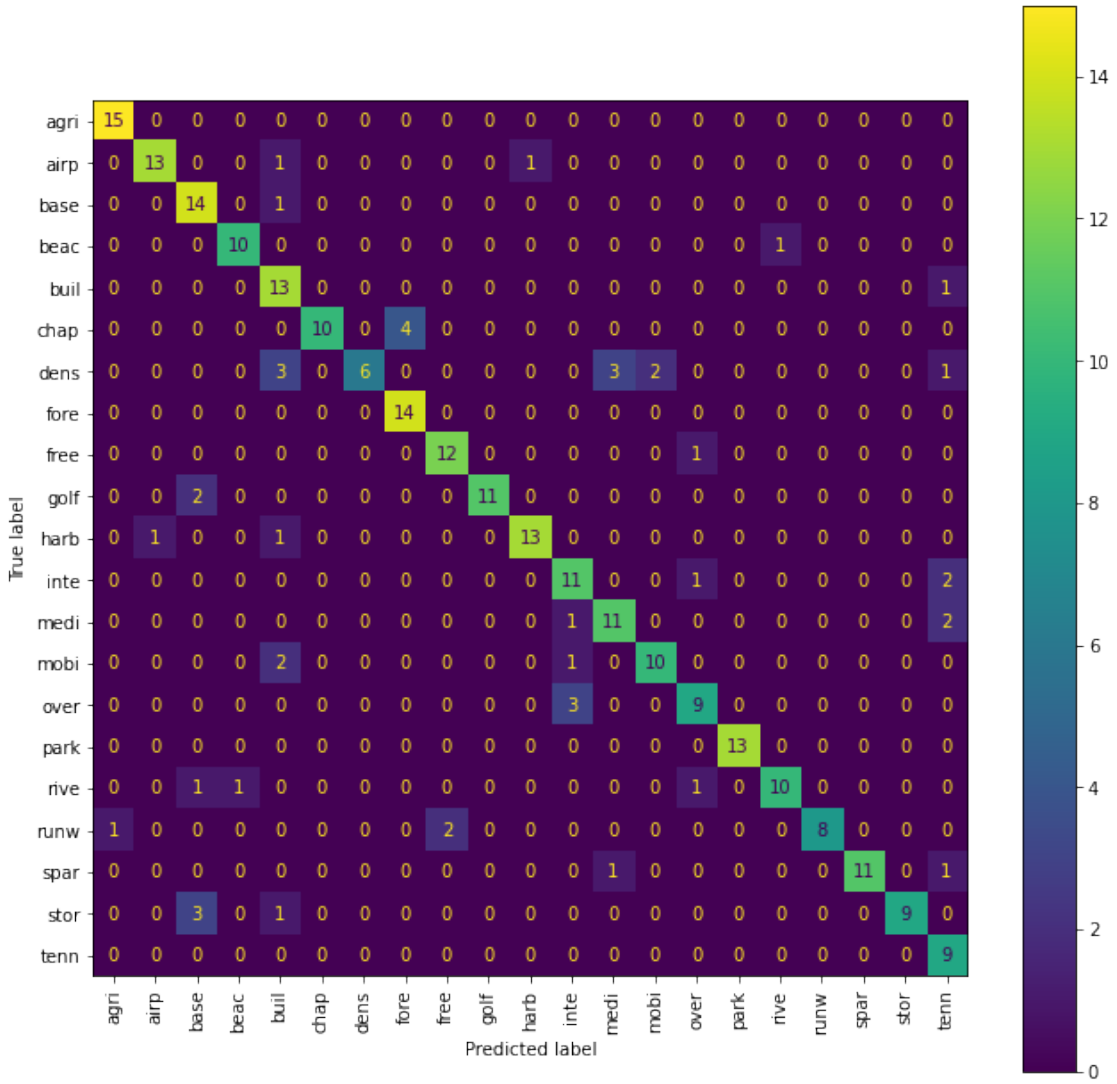


Figure 5. Confusion matrix for transfer learning model. Note that numbers are not normalized for better readability. The average test class contains 13.29 samples.

6.2. Distribution of Predicted Classes.

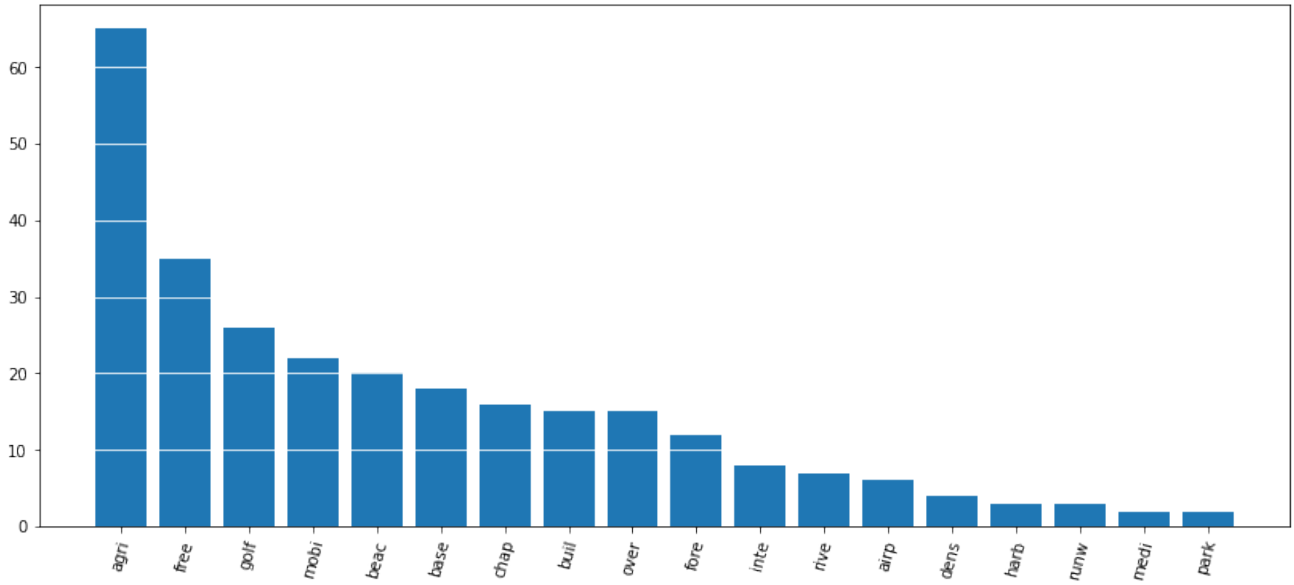


Figure 6. Distribution of predicted classes for BASELINE_V1.

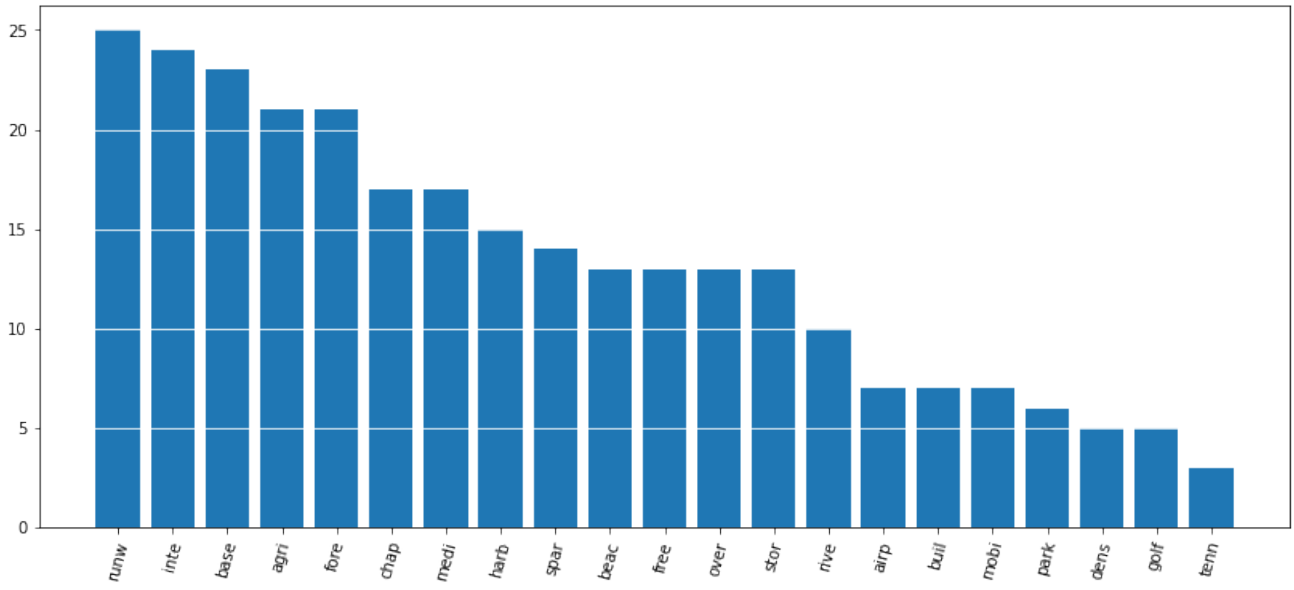


Figure 7. Distribution of predicted classes for BASELINE_V2.

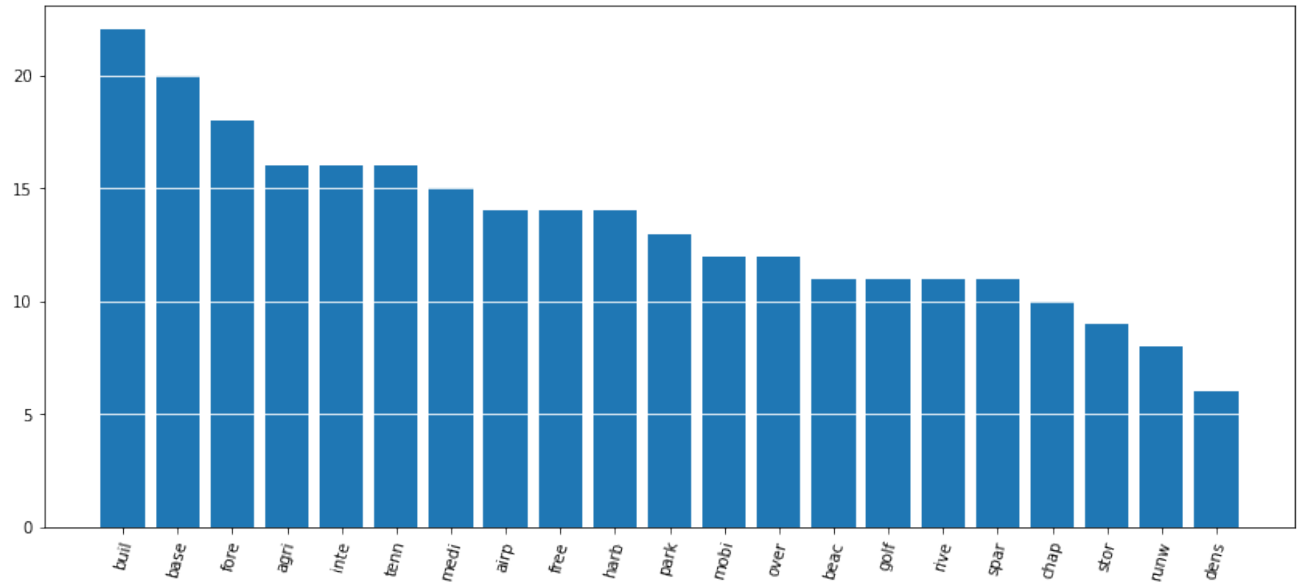


Figure 8. Distribution of predicted classes for transfer learning model.

6.3. Training History.

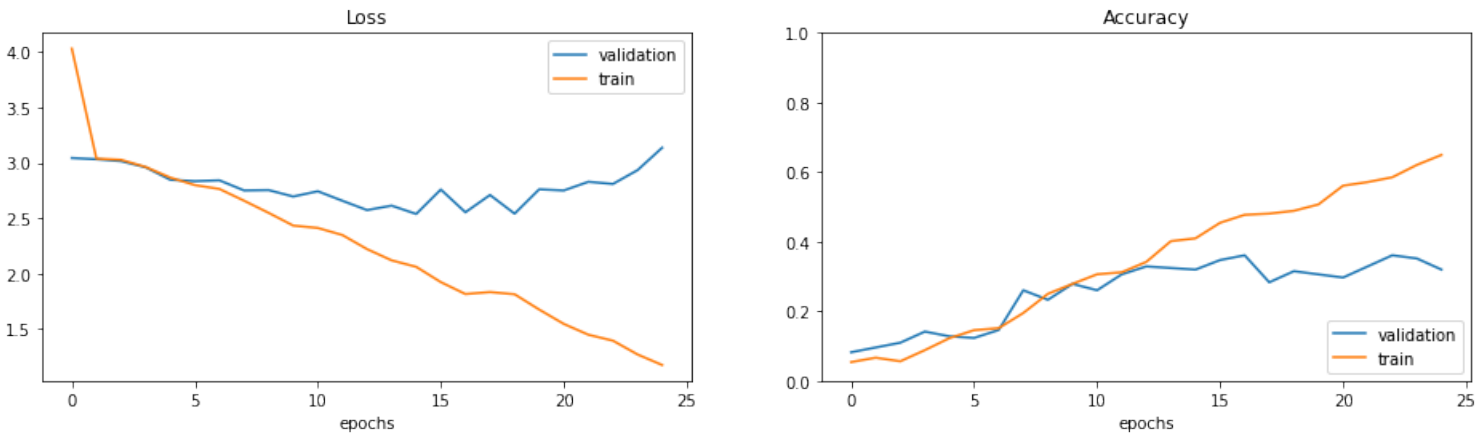
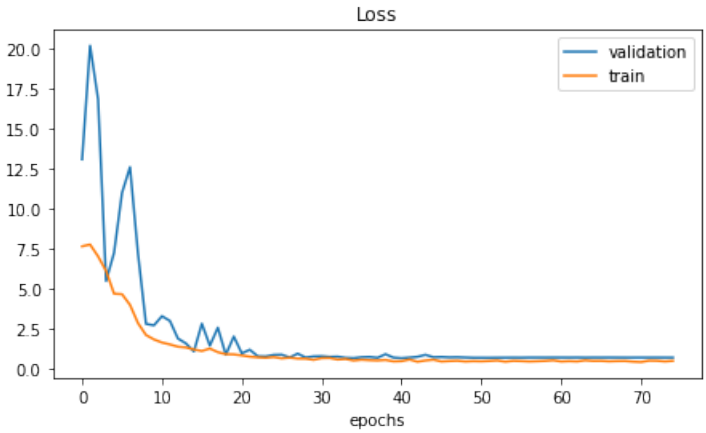


Figure 9. History of training / validation regarding accuracy and loss for BASELINE_V1.



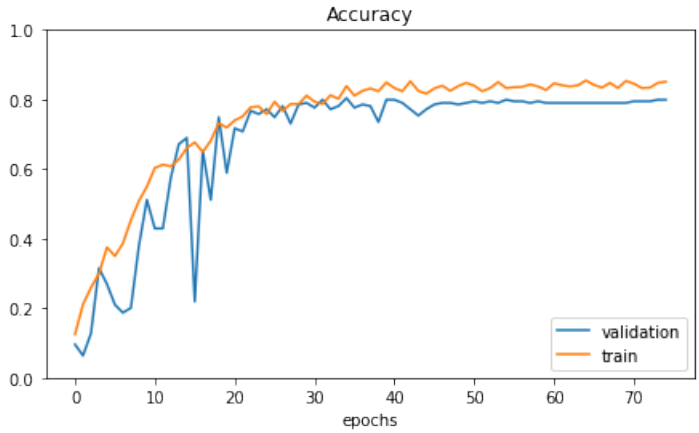
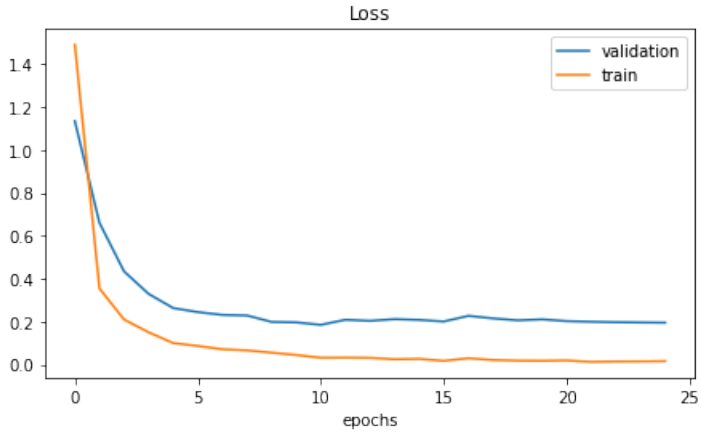


Figure 10. History of training / validation regarding accuracy and loss for BASELINE_V2.



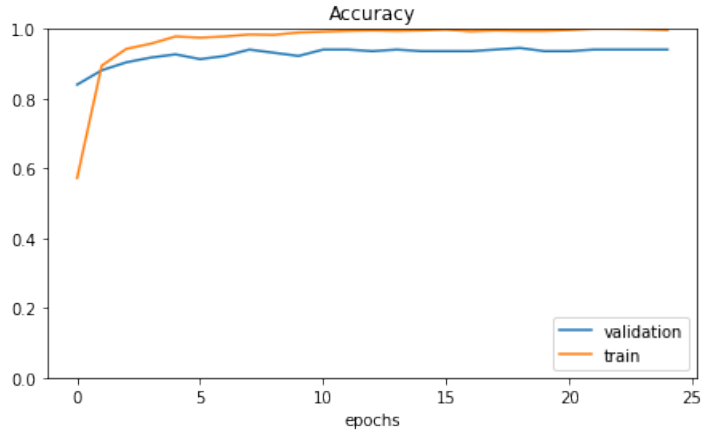


Figure 11. History of training / validation regarding accuracy and loss for transfer learning model.