Task 2: CNN for image classification

I. Introduction

The aim of this paper is to provide an insight into the procedures that have been carried out when addressing an image classification problem.

The problem in mind is regarding 13394 images, split into 10 categories based on the presence of certain objects with them. Such as a golf ball or a musical instrument for instance. The solution implemented in the report involve a convolutional neural network's development and training using the above-mentioned data, whilst evaluating its ability to classify new images accurately. The ultimate goal is to develop a model that can correctly recognize the presence of an object from the 10 trained categories in any given image.

As already mentioned, the machine learning technique implemented in this report is a convolutional neural network. Two variations of such network are going to be used. One using a self-made architecture and the other using transfer learning, applying Mobilenet's model architecture and prior knowledge.

Such image classifications as the ones carried out in this report have a wide range application within numerous industries. For instance, security systems can identify potential security issues, alongside suspicious activities. In healthcare, such classifications can aid the decision making in diagnosis and treatment of patients, by analyzing medical results. It should be noted that for such purposes, much more developed and complicated networks are being implemented, to the ones here.

The results from those various models implemented, had shown that transfer learning, was able to yield the best results, by a great margin relative to the own developed models. However, as outlined later, those results might have been greatly influenced by the choice of transfer learning model and the provided data for the task.

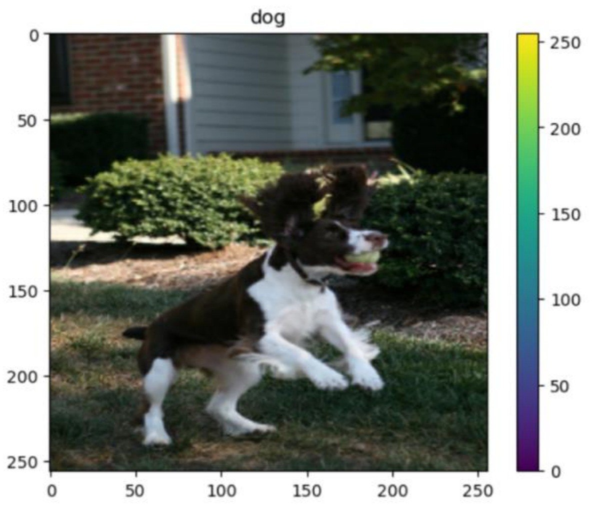
II. Data and Preliminary Analysis

A. *Dataset*

The dataset provided for this report includes a total of 13394 images, which have been divided into training and testing sets by a 70/30 split. Each set consist of 10 classes, with each image being assigned to a class label. To simplify the understanding of the classes, the class labels have been renamed based on the "main" object present in the images, as followed: 'building', 'dog', 'fish', 'gas_station', 'golf', 'musician', 'parachute', 'radio', 'saw', 'vehicle'.

Further it should be noted that the images are not standardized in terms of their height and width. However, they share the same color palette of 3 color channels, implying that the pixel values of the image range from 0 to 255. As it can be seen from the image properties in the following figure:

```
Pixel value range: [0.0, 255.0]
```



B. *Assumptions used throught this report*

Some assumptions are going to be held through the report.

1. Images only belong to one class**.** This would imply that if an image has a few objects into its composition, it can be assigned only to a single class.
2. The structure of a CNN input should follow the one of an image (Teuwen, 2020). As a result, such models are able to use the relationships between pixels in an image at a local level of the image to aid higher-level understanding.
3. Generic knowledge about features on a large enough dataset, can be leveraged onto other datasets (Yamashita, 2018). This encompasses the whole idea behind transfer learning, which uses already learned knowledge from large datasets, onto seemingly unrelated ones.
4. The addition of artificial noise to a dataset, can lead to better model performance (Shorten & Khoshgoftaar, 2019). Implying that an issue of overfitting can be solved through augmentation, leading to improved model fitting.

III. Methods

Convolutional neural networks are a variation of a neural network, which are designed in mind for image classification tasks. In this report there are two main methods of such a network that have been implemented:

- Designing an own architecture of a CNN, where a model is to be trained on provided images and their respective labels. During training all the weights and biases along the network are optimized, in order to accurately classify images.
- Using a technique called transfer learning, where an already trained model with established optimum weights and biases is used to carry out the classification.

The end goal of those two methods is to produce a convolutional neural network, capable of classifying unseen images with a high degree of accuracy.

Regardless of which approach is taken, convolutional neural networks consist of the same fundamental stages and processes. What creates a difference amongst models, would be their architecture, or how those stages and processes are organized. Therefore, it would be worthwhile going over those, before discussing the methods used in this report.

A. *Convolutional neural network processes*

Where this network gets its name from is from the use of convolutional layers. Those represent a set number of learnable kernels which are applied across the input image. As those kernels are smaller than the image, they are "slid" across it and at each step, they compute the dot product between the examined region and the kernel. This process creates feature maps, which are designed to extract a certain feature from the raw input, such as horizontal or vertical lines for instance.

Usually after each convolutional layer, there is an activation layer. Those represent one of the key reasons, of why convolutional neural network have such high learning power. This is because, they bring non-linearity to the model. As when computing the dot product, in the process above, a simple linear transformation is applied, however as it passes through an "activation layer", non-linearity is introduced, allowing for the identification of much more complex relations.

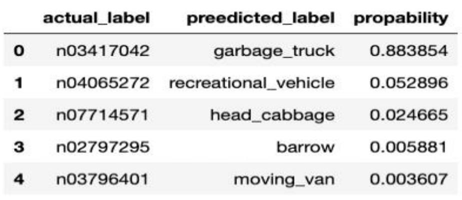
After either a single or multiple convolutional layers with their respective activation functions, a pooling layer tends to be added to the architecture. The primary purpose of such a layer is to reduce the spatial dimensions of the features maps, whilst retaining the key information about various patterns and relationships among them. Therefore, such a layer, returns much simpler and smaller feature maps, which represent where the filter (kernel) did the best in matching the input image. Allowing to aid the identification of the most import features and patterns.

After applying convolutional and pooling layers either in the given order or in various combinations, a dense layer is added afterwards. It serves the purpose to flatten the respective feature maps to a one-dimensional vector. Those are then used as an input to a neural network, where the final layer's results represent the predicted class of the image input.

Backpropagation is where the "learning" process takes place. All the weights and biases in the model, including the learnable kernels mentioned earlier, are initialized with random values. Afterwards the input is processed through the model to obtain an output and the loss function is calculated. Which represents by how much the output differs from the actual "target" values. To optimize the model, all those weights and biases are adjusted using gradient descent, which minimizes the gradient of the loss function, in order to find the optimal values for the parameters in mind. By repeatedly adjusting all the weights and biases through backpropagation, the model is able to "learn" the features of the input and achieve an optimal classification result.

B. *Transfer learning*

Another technique is also going to be implemented, to tackle the classification problem at hand. It is called transfer learning, which involves using a pre-trained convolutional neural network as an initial ground for our task. It is important to note that simply applying the pre-trained model to the testing images would not be appropriate, as the pre-trained model might have been trained on completely different class labels, to the ones we have in mind. For instance, in our case, we would want to identify this image as having a vehicle displayed in it, and not garbage truck or recreational vehicle, as MobileNet has done so.



	actual_label	preedicted_label	propability
0	n03417042	garbage_truck	0.883854
1	n04065272	recreational_vehicle	0.052896
2	n07714571	head_cabbage	0.024665
3	n02797295	barrow	0.005881
4	n03796401	moving_van	0.003607

Therefore, such a model must be adapted, to be relevant for the classification problem at hand. This is usually achieved by removing the final dense layers, as those are responsible for the final details of the classification result. And replacing them with dense layers applicable to our problem. This would allow to gain the knowledge of the pre-trained model and apply it to a task in mind. Supposedly providing the benefits of improved accuracy, efficiency, computational power required.

IV. Experiments

In this section of the report, the six experiments that were conducted will be discussed. Four of those, were carried out using an own developed convolutional neural network architecture and the other two used the MobileNet's architecture to implement transfer learning.

It should be noted that for all experiments, the image datasets were imported using an uniform size of 224 x 224. This was influenced due to first of all suiting all the architectures used, but also because MobileNet , has been originally trained on images of the same dimensions.

Further for all experiments, the training dataset was split into 80/20 ratio for training and validation respectively. This was chosen to better evaluate the performance of the models and detect the presence of overfitting during the training stage.

A. *Own architecture experiments*

Before delving into the architecture design, it is important to outline that all of the pixel values for the upcoming tasks in part A, have been normalized between 0 and 1. So that the impact of variations in light and color are minimized.

1) *Baseline own architecture*

Initially a simple architecture has been developed, which can act as a basis for all the subsequent experiments that will be carried out. In order to examine, whether the taken measures have led onto any improvements in the performance of the convolutional neural network.

The architecture is as followed:

- Convolutional layer with sixteen 3 x 3 kernels, followed by a ReLU activation function. Such activation function implies that only the positive values will be returned, whereas the negatives will be substituted with zeros.
- Pooling layer of 2 x 2 size.
- Two subsequent convolutional layers with thirty-two 3 x 3 kernels, followed by ReLU activation functions.
- Pooling layer of 2 x 2 size.
- Two subsequent convolutional layers with thirty-two 3 x 3 kernels, followed by ReLU activation functions.
- Pooling layer of 2 x 2 size.
- Flatten layer, which converts the previous pooling layers into one dimensional vector.
- Dense layer with 256 nodes, followed by a ReLU activation function.
- Dense layer with as many nodes as classes in our classification task. Softmax activation function is used in order the output values to be transformed in the range between 0 and 1, whilst the sum of all output values to equal 1. Effectively giving us the predicted probabilities for each class. In order for those results to be turned into a class label, they are run through another function, known as Argmax.

The optimizer of choice is Adam. It has been chosen over a Stochastic Gradient Descent (SGD), due to its faster convergence and better performance (Kingma & Ba, 2014).

Categorial cross-entropy loss function was chosen, as it is appropriate for problems that involve the classification of more than two classes.

2) *Implementing performance enhancers*

i) Data augmentation

The training dataset will be artificially increased in size by the implementation of data augmentation. This involves applying various transformations to the image dataset in mind, in order to provide the model with additional instances of a given class. Supposedly, allowing the model to perform better on unseen data.

It should be noted that the augmentation is applied only to the training dataset, as the validation and testing ones, are solely used for evaluative purposes. Thus, introducing transformed data within those evaluators, may result in wrongly classified outputs, thus negatively offsetting performance.

ii) Batch normalization and Dropout Layers

Batch normalization will be introduced into the architecture of the model. As it ensures each layer that is applied, will be standartized by having a mean of zero and a variance of one. Allowing to increase the training efficiency. Further it should allow for improving the model's generalization abilities, leading to better classification results.

Also as training of the model takes place, some nodes may become too reliant on previous layers, leading to assigning them a higher weight in terms of importance. This may learn the model "too well" on the training data, so that it cannot perform well on unseen data. Dropout layers can be introduced to tackle this issue, by randomly "disabling" nodes during training thus reducing the chance of overfitting.

iii) Joint impact of performance enhancers

Lastly an experiment will be carried out to examine, the joint impact of batch normalization, dropout layers and data augmentation.

B. *Transfer Learning implementation*

For the experiments explained below, before running the models, the pixel values have been normalized in the range between -1 and 1. This has been done out of convenience, as keras already provides a pre-processing function for MobileNet that does so.

1) *Retaining all the prexisting weights and biases*

Firstly, the MobileNet model was implemented, which has already been trained on the ImageNet dataset. This implies that model's weights and biases have already been established at their optimum levels. Therefore, it would be beneficial to make the neural network non trainable, so that those parameters can leverage all the prior learned knowledge for our classification task in mind. As already outlined in the methodology section, in order for such result to be achieved, few of the last layers, have to be replaced with our own dense ones. So that transfer learning can take place, with regards to our class labels.

2) *Fine tuning the model*

Further another approach for customizing a pre-trained model, known as fine-tuning, will be implemented. It is a follow up to the transfer learning covered in the point above. As it involves setting some of the already pre-trained layers, available for training. With the idea to allow the model in mind, to better adapt to the task.

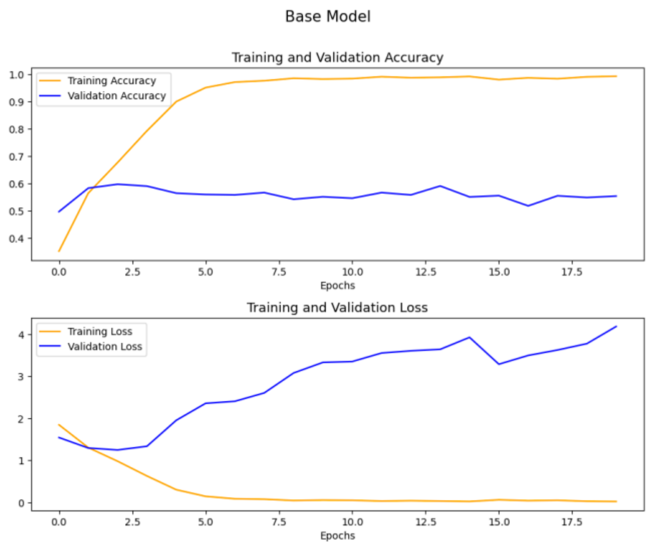
V. RESULTS

All the results that will be discussed, would be regarding model training taking the span of 20 epochs. The main motivation behind this choice is the computational resources available at the time of writing this report.

A. *Own architecture results*

1) *Base experiment*

In the first experiment, the base architecture was evaluated. As it can be seen on the figure below, it was able to achieve a very high level of training accuracy, at almost 100%. However, this is not an indication of a good model performance. Because as it can be seen, there is a constant significant deviation between the validation training and loss curves from their respective training ones. This may be indicating an overfitting issue, where the model specializes on the training set and is not being able to perform well unseen data.



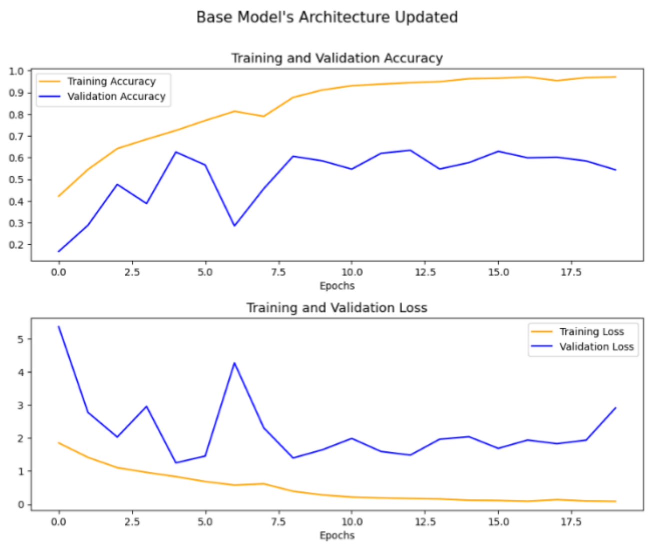
The results from test accuracy and test loss, are 56% and 4.08 respectively, which support the overfitting theory. As they show that the model is not able to accurately classify a great number of images in the test dataset, whilst not being confident in doing so. A F1 score of 55% further supports that statement.

In appendix 1a) is provided a confusion matrix, which visualizes the convolutional neural network's classifications on the test set for each of the ten classes in mind. Its results keep consistent with the result outlined above.

In appendix 1b), nine random images from the test dataset have been plotted along with their respective actual and predicted labels. As it can be seen only 5 out of 9 images are predicted correctly, which amounts to 55,5%. Further outlining the poor performance of the model, although on a relatively small sample size.

2) *Dropout layers and batch normalziation*

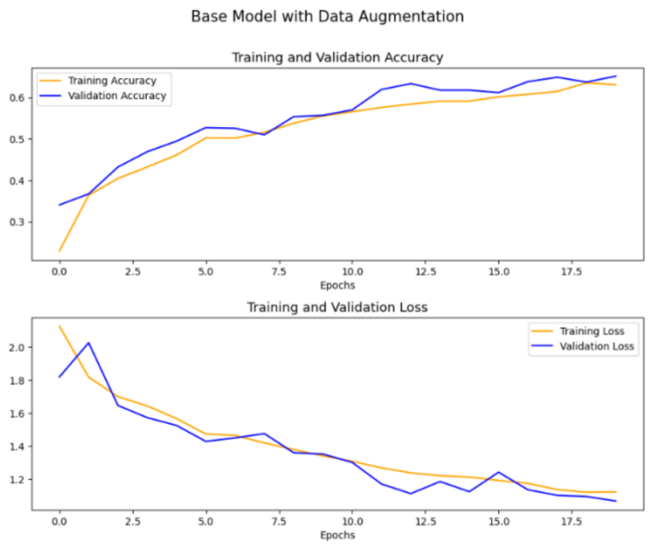
In the second experiment, batch normalization and dropout layers have been introduced to the base architecture of the model. However, their implementation, did not give us the expected remedy of reducing overfitting. As it can be seen on the figure below, the gap between the training and validation accuracy and losses, visually appears reduced by a small margin, perhaps due to the uneven shapes of both validation curves. However, that gap remained of a quite noticeable size. Suggesting that overfitting has not been tackled.



The evaluation of the model on the testing set, confirmed our suspicions. As the model's accuracy experience a relative decrease to the first experiment, sitting at 54%. Suggesting that this model is only slightly better than if a random guess was taken for the classification task. The test loss remained at a high level of 3, further suggesting that this model is also not confident in its classifications.

3) *Data augmentation*

In the third experiment, data augmentations was applied to the training set, which was used on the base convolutional network architecture. The result was pleasing, as at first glance, as this remedy was able to reduce the extend of overfitting. This can be examined from the graph bellow, as the training and validation accuracy and loss lines, moved much closer to each other, compared to the previous two examined cases. However, it should be noted especially, on the accuracy curve, that the model was not able to achieve high performance, such as into the eighties and nineties percental range.



Those results are further supported by the test loss and accuracy measures. Which indicate that the model was able to achieve 66% test accuracy and a test value of 1.05, a great improvement relative to the previous experiments. Such results are suggestive that the model was much better at capturing the generalized trends from the train data, rather than overspecializing on it. And thus, its classifications were carried out with much higher confidence.

In appendix 2, a high overview of that trend can be examined from the confusion matrix provided for this experiment. The improvements that data augmentation has brought become much more apparent when compared to the earlier confusion matrix provided.

4) *Dropout layers, batch normalziation and data augmentation*

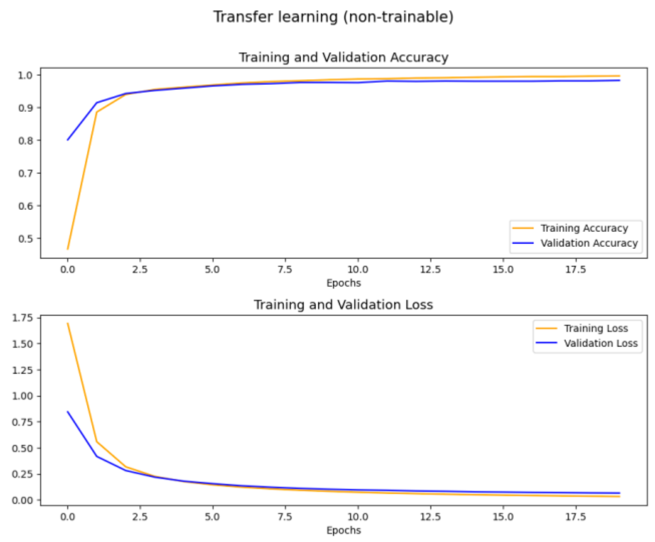
In this experiment, all the methods above were combined, to examine whether their joint effect will be greater to this on their own. However not a lot of time and effort will be devoted to this subsection, as the achieved results were a decrease in performance, relative to the ones examined with only dropout layers and batch normalization. As test accuracy decreased to 51% and test fall to 1,7.

Perhaps it is interesting the test loss fell, whilst accuracy did not increase. This implies that the model has gained much more confidence in its classifications, however this confidence is not met by a proportionate increase in accuracy.

B. *Transfer learning results*

1) *Non - trainable weights and biases*

In sharp contrast to the previous experiments, the implementation of MobileNet, was able to deliver great results. As it can be seen from the plot below, both validation and testing accuracy and loss, are overlapping after just a few epochs. Suggesting that the model was able strike balance, between generalizing the extracted patterns from the training data, but also providing the right amount of detail to those generalizations.

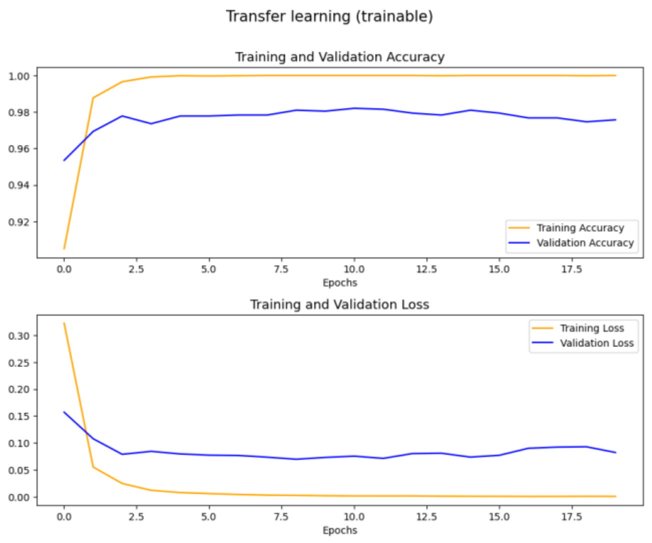


Those results are further confirmed, by evaluating the model's performance on the test dataset. Here the models, was able to show 98% of accuracy and 0.06 test loss. Indicating the model is able to make really accurate classifications, with a great amount of confidence. This further can be seen from the confusion matrix visualizing the classifications made provided at appendix 3.

2) *Fine tuning the model*

In this part it will be looked over, whether the results from above, can be improved by making the last 24 layers of the model trainable.

However, the accuracy and loss plots below, indicate that a step backwards was rather taken. As signs of overfitting became apparent, represented by the widened distance between the validation and training curves. However, it can be argued that such a model is still much better to those proposed in section A. Due to the accuracy curves being in the 95th percentile and the loss curves being below 0,15.



This is further proven by the performance tests carried out on the test dataset. For which test accuracy was at 97,9% and test loss 0,07. Suggesting that the model is still very good in terms of its predictive power and confidence of its output. However, it should be noted that it is slightly worse performing to the non-trained transfer learning model. In appendix 4, the confusion matrix for this experiment can be found.

It should be mentioned that the results gotten in this part of the section, might have been influenced by the dataset provided for the classification task. As the images used, are extracted from the ImageNet dataset, on which the MobileNet has been trained. Therefore, the pre-trained parameters of the model might have been perfectly fitting to the provided training data. Implying that there is uncertainty surrounding whether those results could be replicated using another image test dataset.

VI. Conclusive Remarks

In this report there were two main methods used to find the optimal convolutional neural network technique for a classification task, containing 10 class labels. The results, showed that transfer learning yielded the better model and in specific transfer learning for which all layers were made non trainable. Further, the own-architecture models, showed that perhaps their design had been inherently flawed, as all of the experiments had the tendency to overfit to the training set and thus led to poor performance. Nevertheless, this technique demonstrated the benefits, which data augmentation may bring.

Going forwards, it may be worthwhile examining other various transfer learning models and own architectures, which may bring even greater classification performance. Further it would be valuable to examine different set of optimizers and loss functions, alongside the addition of extra epochs, so that the models could be evaluated to a greater extend.

References

1) Kingma, D. P. & Ba, J. 2014, 'Adam: A Method for Stochastic Optimization', arXiv preprint arXiv:1412.6980, viewed 16 March 2023, https://arxiv.org/abs/1412.6980.

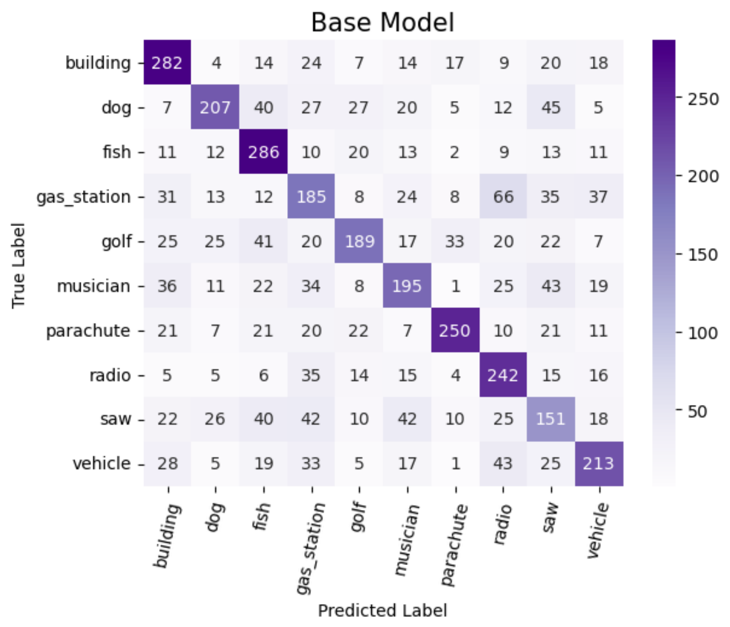
2) Shorten, C & Khoshgoftaar, TM 2019, 'A survey on image data augmentation for deep learning', Journal of Big Data, vol. 6, no. 1, pp. 1-48, viewed 16 March 2023, https://doi.org/10.1186/s40537-019-0197-0.

3) Teuwen, L 2020, 'Convolutional neural networks for image classification', Towards Data Science, viewed 16 March 2023, https://towardsdatascience.com/convolutional-neural-networks-for-image-classification-88ca951f9345.

4) Yamashita, R 2018, 'Transfer Learning and Image Classification using Keras', Towards Data Science, viewed 16 March 2023, https://towardsdatascience.com/transfer-learning-and-image-classification-using-keras-on-kaggle-kernels-c76d3b030649.

Appendix

1)A) Confussion Matrix of base model



	building	dog	fish	gas_station	golf	musician	parachute	radio	saw	vehicle
building	282	4	14	24	7	14	17	9	20	18
dog	7	207	40	27	27	20	5	12	45	5
fish	11	12	286	10	20	13	2	9	13	11
gas_station	31	13	12	185	8	24	8	66	35	37
golf	25	25	41	20	189	17	33	20	22	7
musician	36	11	22	34	8	195	1	25	43	19
parachute	21	7	21	20	22	7	250	10	21	11
radio	5	5	6	35	14	15	4	242	15	16
saw	22	26	40	42	10	42	10	25	151	18
vehicle	28	5	19	33	5	17	1	43	25	213

B) Image classification of base model

Base Model

Actual label:parachute;
Predicted label:radio

Actual label:fish;
Predicted label:fish

Actual label:saw;
Predicted label:saw

Actual label:golf;
Predicted label:golf

Actual label:gas_station;
Predicted label:gas_station

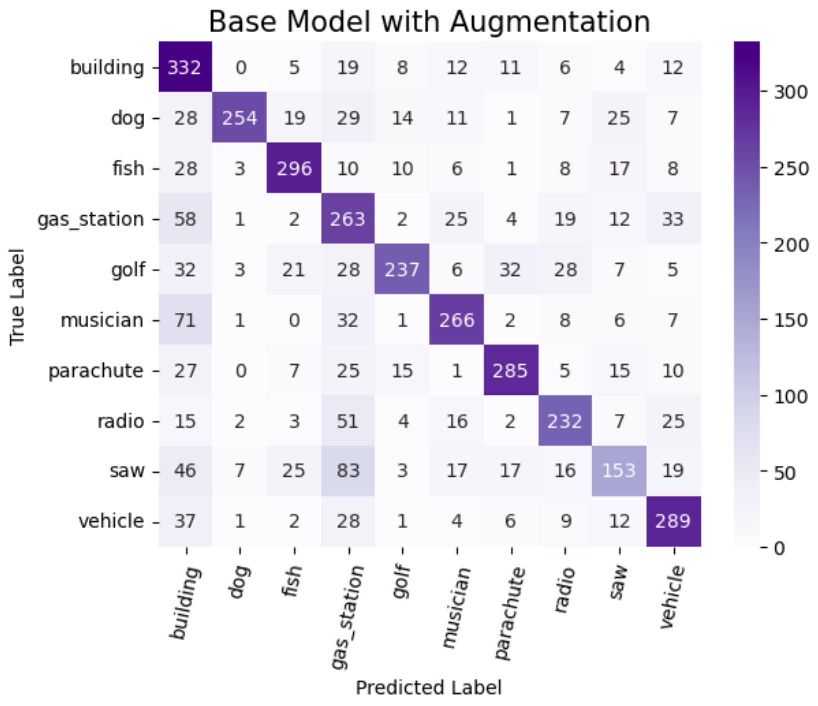
Actual label:fish;
Predicted label:vehicle

Actual label:musician;
Predicted label:dog

Actual label:musician;
Predicted label:musician

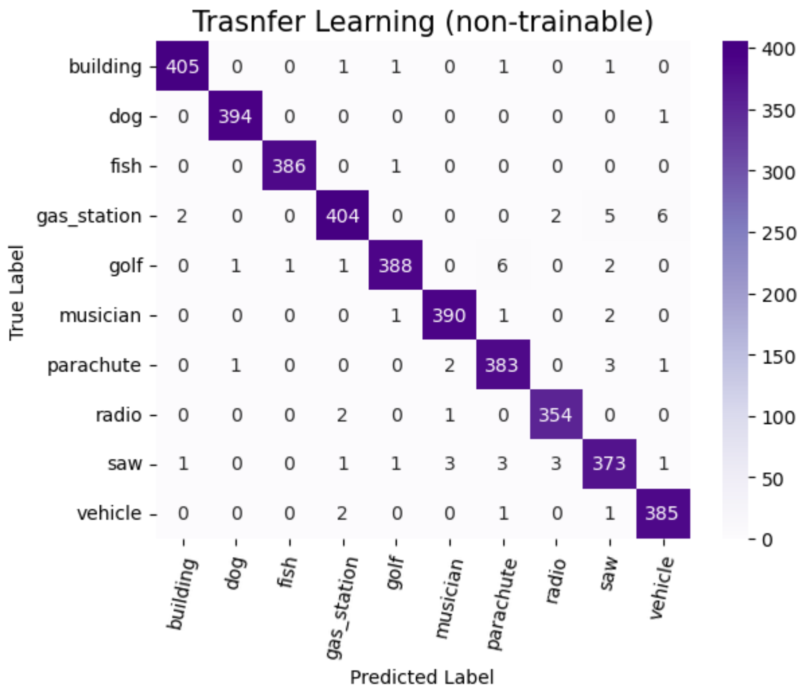
Actual label:building;
Predicted label:musician

2) ***Confussion Matrix of base model with augmented training data***



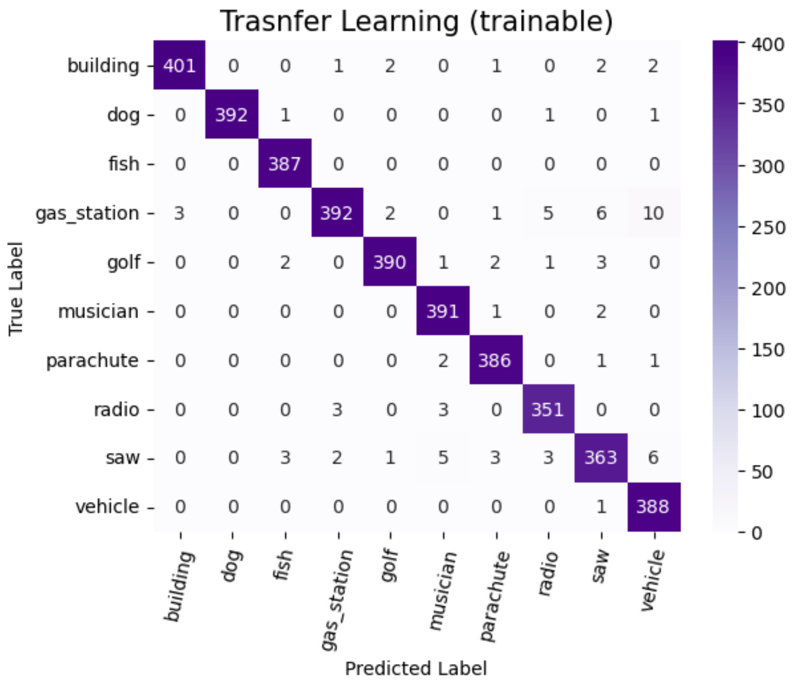
True Label \ Predicted Label	building	dog	fish	gas_station	golf	musician	parachute	radio	saw	vehicle
building	332	0	5	19	8	12	11	6	4	12
dog	28	254	19	29	14	11	1	7	25	7
fish	28	3	296	10	10	6	1	8	17	8
gas_station	58	1	2	263	2	25	4	19	12	33
golf	32	3	21	28	237	6	32	28	7	5
musician	71	1	0	32	1	266	2	8	6	7
parachute	27	0	7	25	15	1	285	5	15	10
radio	15	2	3	51	4	16	2	232	7	25
saw	46	7	25	83	3	17	17	16	153	19
vehicle	37	1	2	28	1	4	6	9	12	289

3) ***Confussion Matrix of non-trainable transfer learing***



True Label \ Predicted Label	building	dog	fish	gas_station	golf	musician	parachute	radio	saw	vehicle
building	405	0	0	1	1	0	1	0	1	0
dog	0	394	0	0	0	0	0	0	0	1
fish	0	0	386	0	1	0	0	0	0	0
gas_station	2	0	0	404	0	0	0	2	5	6
golf	0	1	1	1	388	0	6	0	2	0
musician	0	0	0	0	1	390	1	0	2	0
parachute	0	1	0	0	0	2	383	0	3	1
radio	0	0	0	2	0	1	0	354	0	0
saw	1	0	0	1	1	3	3	3	373	1
vehicle	0	0	0	2	0	0	1	0	1	385

*4) **Confussion Matrix of trainable transfer learning***



	building	dog	fish	gas_station	golf	musician	parachute	radio	saw	vehicle
building	401	0	0	1	2	0	1	0	2	2
dog	0	392	1	0	0	0	0	1	0	1
fish	0	0	387	0	0	0	0	0	0	0
gas_station	3	0	0	392	2	0	1	5	6	10
golf	0	0	2	0	390	1	2	1	3	0
musician	0	0	0	0	0	391	1	0	2	0
parachute	0	0	0	0	0	2	386	0	1	1
radio	0	0	0	3	0	3	0	351	0	0
saw	0	0	3	2	1	5	3	3	363	6
vehicle	0	0	0	0	0	0	0	0	1	388