Classification of cats and dogs images into breeds with fine tuned CNNs

Final project report on Machine Learning and Deep Learning (650.025, 23W), University of Klagenfurt

Jerzy Pawlik*
Poznań University of Technology
Student ID: 12237552 (Erasmus)
05.02.2024, Klagenfurt, Austria

***Abstract*—Classification of cats and dogs is a common problem on which convolutional neural network architectures have been tested. Simply distinguishing whether an image shows a cat or a dog is not a challenge for humans. Consequently, the practical applications of such classifiers are quite limited. However, accurately classifying a cat or dog into its specific breed is more demanding manually, requiring knowledge of all the breeds or searching through examples of each breed every time we want to classify an image. Additionally, similar architectures and training procedures can be applied to solve more serious issues, such as classifying plant diseases.**
In this study, two neural network architectures, ResNet101 and InceptionV3, are applied to classify cats and dogs into 37 different breeds. The achieved accuracies on the test set are 92,4% for ResNet101 and 87% for InceptionV3. Both neural network models were PyTorch implementations pretrained on Image net and fine-tuned on purpose of this study.

I. Introduction

This study is a final project on Machine Learning and Deep Learning university course which among other deep learning topics covers the convolutional neural networks. The motivation behind it was to test the behaviour of CNNs in solving a practical problem, try to improve the performance and compare the results with literature and other solutions. Classification between cats and dogs or their breeds is commonly used to test different CNN architectures. Distinguishing between cat and dog is a simple binary classification problem, but classifying different breeds is part of a more complicated multi-class single-label supervised learning category. Problems like automatically detecting plant diseases on images [1], to prevent them from spreading and thus increasing world food security, are also part of this category. The same neural network architectures can be also used there, and training procedure is very similar. From the models analysed in related work considered in this study, two CNNs (ResNet101 and Inception v3) were selected based on their performance. The obtained accuracies of 92,4% and 87% respectively were comparable with those in related studies. For the training, a commonly used technique called "transfer learning" was applied, which is utilizing pretrained models and substituting their top layers to recognize new categories. This allows to save computational resources. Models that are considered here were both pretrained on ImageNet which is a vast dataset consisting of 14 197 122 images grouped into roughly 21,000 categories. The dataset used for training testing and validation was Cats and Dogs Breeds Classification Oxford Dataset [2] downloaded from Kaggle, consisting of 37 different breeds of cats and dogs. Moreover a few custom augmentations were applied to the training data

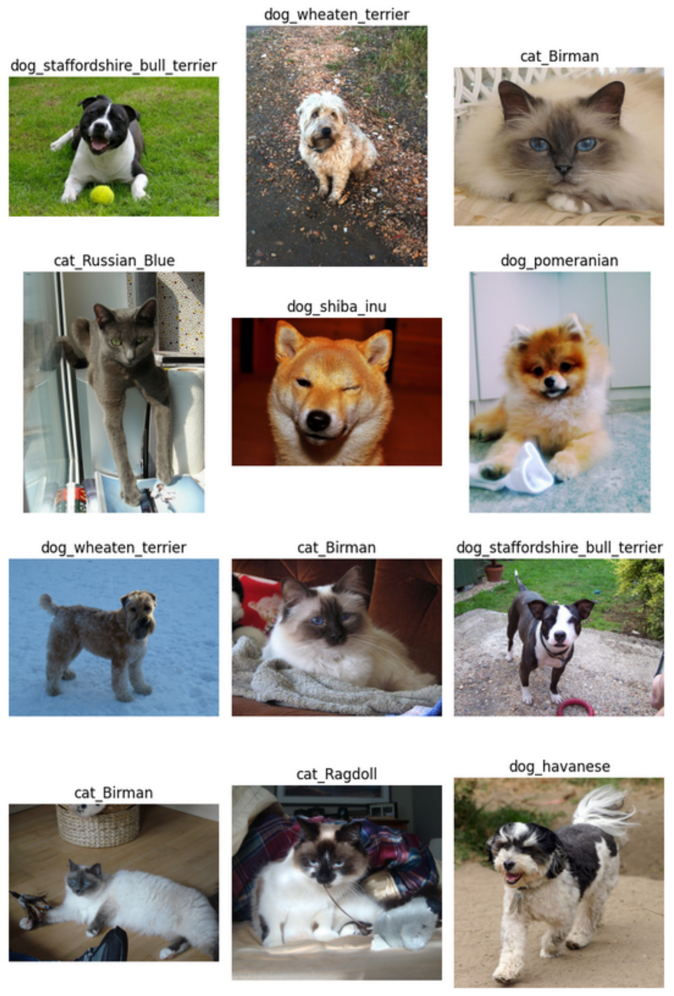


Fig. 1. Sample images from the dataset

II. Data

The dataset is balanced and contains 7349 images saved in a jpg format, what is roughly 200 images per class. Total size of it is 821.05 MB, what makes it medium size according to Kaggle. Images are annotated with class ID (from 1 to 37), a digit indicating whether it is a cat or a dog, and breed ID which are digits from 1 to 25 or from 1 to 12 for dogs and cats respectively. The filename of each jpg file starts from a breed name, which is the same for all the animals with the same class ID. Image resolutions are different, but for the majority, width and height are between 200 and 500 pixels. Most of them have rectangular shapes, what is characteristic for the photos taken by camera (see figure 2 and 3). Animals are photographed in a different poses, from a different distance, perspectives and in a different lightning conditions (see figure 1). The dataset was split into training, validation and test set in proportions 0.64 x 0.16 x 0.2 respectively.

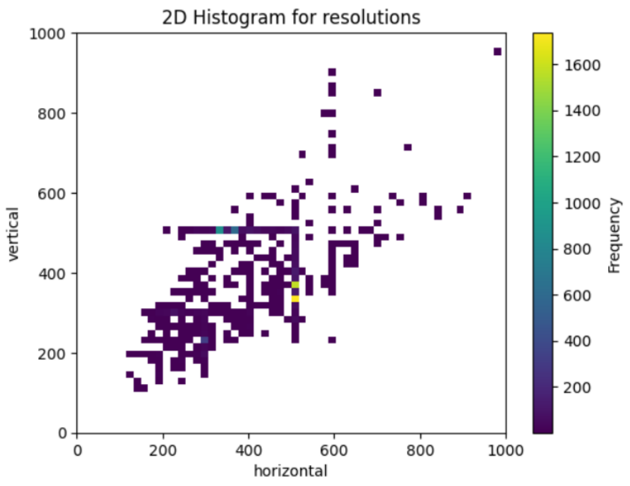


Fig. 2. 2D histogram of image resolutions

III. Related Work

At the time when this project was done, a total of 21 notebooks had already been posted with the dataset. In one of them, user Meet_Vora [3] had obtained 87% accuracy on classifying cats and 89% on classifying dogs. However, classification was done for cats and dog breeds separately, so the model had to handle a smaller number of classes each time. In his approach, the author used fine-tuned keras pretrained MobileNetV2 implementation. He also augmented the dataset by adding transformations like, zooming, random cropping, flipping, rotating, and stretching to already given images. This technique is used to prevent overfitting or provide more data for training to overcome insufficient training data problem. It has been proven empirically by Poojary et al. [4] in 2020 that it actually improves the performance. In the study, he utilized transfer learning to train Keras VGG16 and ResNet50 models to recognize cat and dog images species, once with and once without augmentations. The models trained on augmented

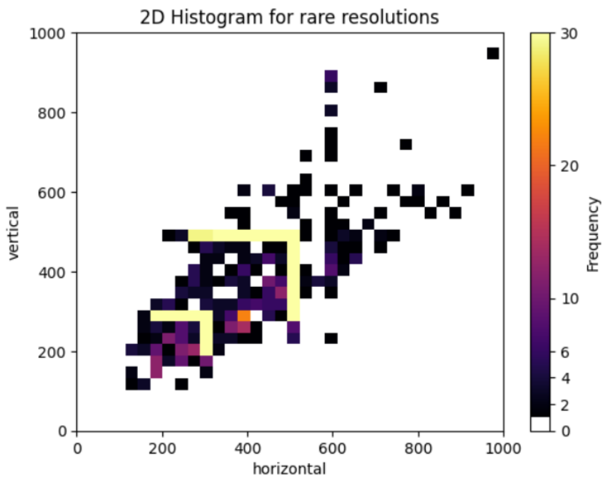


Fig. 3. 2D histogram of the most frequent image resolutions. For a better precision the the colours scale is cut at frequency = 30, what means that all the graph regions where more than 30 resolutions of different dataset images images fit will have a color corresponding to 30 fitting images

dataset achieved significantly better test accuracies, and also differences between accuracies on a training data and testing data were smaller for them. ResNet50 architecture has also outperformed VGG16 by 2% in each case. The transformations that were used to augment the dataset were horizontal flipping, rotation up to 30 degrees, and scaling or zooming. Different types and methods of data augmentation have been well visualized and described by user Ishan Dutta in a Kaggle notebook [5], which was useful resource for studying the topic of data augmentation for this project.

Furthermore, there are additional studies conducted by senior university students or scientists, which compare the performance of various pre-trained CNNs in classifying between cats and dogs or different breeds of cats and dogs. These studies utilize diverse datasets, varying in the number of examples and the quality of images, as well as different training procedures. In 2020, Mahardi et al. [6] fine-tuned two pre-trained VGG architectures (VGG19 and VGG16) on a dataset consisting of 20,574 images of cats and dogs to classify them into 20 breeds. The achieved accuracies on the testing set were 84.07% for VGG19 and 83.68% for VGG16. In 2019, Borwarnginn et al. [7] achieved better results in classifying images of dogs into 20 selected breeds. They applied the Inception V3 architecture with transfer learning and obtained an accuracy of 96.75%. Additionally, they tested pre-trained NASNet MobilenetV2 and InceptionV3 for classifying dogs into 133 breeds, achieving accuracies of 89.6%, 89.5%, and 91%, respectively.

Moreover, they explored different conventional methods for the classification into 20 breeds. The best-performing method was histogram of oriented gradients with support vector machines, which achieved an accuracy of 79%, although significantly lower than InceptionV3. Similarly, in the study conducted by Bang, Yan Liu, and Kai Zhou [8], the authors compared conventional methods with CNN-based methods and SIFT for classification between cats and dogs. In this work, support vector machines were trained for classification, and CNNs were trained for feature detection. Despite the simpler task, SIFT+SVM achieved only 71% accuracy, while CNN+SVM reached 94%.

As it was mentioned before in the introduction section, problem like classification of plant diseases based on plant images belongs to the same category as the problem considered in this study. Aline Dobrovsky who completed this course last year had trained different CNN architectures for that specific task [1]. All of them achieved accuracies over 96%, but the best performing model was ResNet101, with more than 98% accuracy. Other considered models were DenseNet101 and ConvNext-Base. However, the number of classes and the size of the dataset in that task was greater than here, with over 76,000 images belonging to 88 classes and total size of 17.6 GB. The dataset was unbalanced and also during the training, data augmentation was done.

In summary, the performance of different models varied depending on several factors, such as the training data, training procedure, applied augmentation techniques, or lack thereof [4], and the complexity of the problem [7]. However, it appears that VGG16 and VGG19 architectures [6] exhibit lower performance compared to other mentioned networks, even when dealing with more complex problems, especially when compared to ResNets with 50 or more layers [4] [1]. InceptionV3 also showed slightly better performance than other alternatives [7]. Additionally, practical evidence [4] indicates that data augmentations have a positive impact on increasing model performance. Apart from CNNs, conventional methods have also been applied to solve tasks similar to the one considered in this project, albeit with significantly lower performance [7] [8]. Therefore, based on the gathered knowledge, utilizing InceptionV3 and ResNet101 architectures alongside data augmentation seems to be a reasonable choice. It's worth noting that ResNet101 was applied to a problem of a much larger scale and is significantly more complex compared to VGGs, which may increase the risk of overfitting.

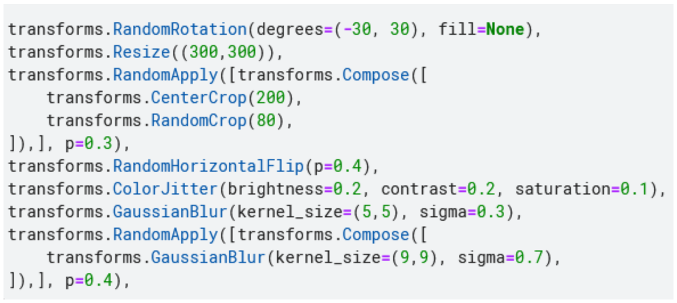
IV. Methodology

The entire training loop and models were implemented using PyTorch. The runs were conducted in a Kaggle environment, utilizing a Tesla P100 graphics card for parallel computations, which significantly speeds up the entire process. The code was written in a way that allows for online training even when the session is closed. Checkpoints were saved during the process to allow for resumption in case of any issues. Additionally, a number of metrics were plotted after every batch and epoch to monitor the procedure. The weights and biases platform was used for automatic tracking of the metrics and configuration of every run.

A. Augmentation and transformations of training data

The purpose of adding transformations to the images loaded into the model is to augment the dataset with more varied examples and improve the model's generalization capabilities by focusing only on relevant features. To achieve this, Gaussian blurring was applied to reduce noise. Images in the dataset were taken from various distances, so to make the classifier more robust, random cropping was applied to the center part of the image. Initially, the center was cropped, and then a random smaller rectangle was cropped from it to avoid missing animals that were typically located in the center of the images. Stronger Gaussian blurring was also randomly applied. Additionally, color jitter for brightness, contrast, and saturation, as suggested in [1], was applied to make the model robust to different cameras and lighting conditions. All transformations used by Poojary et al. [4] were also implemented here.

Since the images have different resolutions, resizing was applied before the entire process. Finally, the images were resized to the input format of the neural network, transformed into tensors, and normalized. The implementation of all transformations is shown in the figure 4.



```
transforms.RandomRotation(degrees=(-30, 30), fill=None),
transforms.Resize((300,300)),
transforms.RandomApply([transforms.Compose([
    transforms.CenterCrop(200),
    transforms.RandomCrop(80),
]),], p=0.3),
transforms.RandomHorizontalFlip(p=0.4),
transforms.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.1),
transforms.GaussianBlur(kernel_size=(5,5), sigma=0.3),
transforms.RandomApply([transforms.Compose([
    transforms.GaussianBlur(kernel_size=(9,9), sigma=0.7),
]),], p=0.4),
```

Fig. 4. Python implementation of custom transformations

B. Model selection

Both of the selected architectures are powerful convolutional neural networks with many layers. IncpetionV3 has 42 layers and ResNet101 has 101 layers. Inception characterises in utilizing series of layers called "Inception modules" within the architecture. Each Inception module consists of multiple convolutional layers with filters of different sizes applied in parallel. These parallel paths capture features at different scales and resolutions, and the outputs from these paths are then concatenated or merged before being passed to the next layer in the network. This multi-path approach enables the network to extract diverse and rich hierarchical features from the input image, enhancing its ability to understand and classify objects effectively. In the architectures with many layers, vanishing gradients may become a huge challenge. ResNets address it by introducing skip connections, within the network architecture. Instead of relying solely on the forward flow of information through consecutive layers, ResNets allow information to bypass a few layers and be directly fed to deeper layers. This way, the original input signal can flow more easily through the network, mitigating the problem of vanishing gradients. Also, InceptionV3 by introducing parallel connections in its inception modules, mitigates this problem.

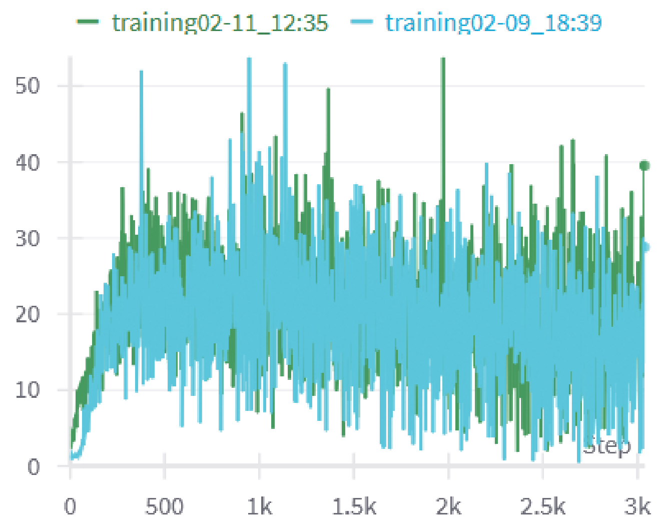


Fig. 5. Gradient magnitude every bath. Blue line stands for training the ResNet101 and the green one for TnceptionV3

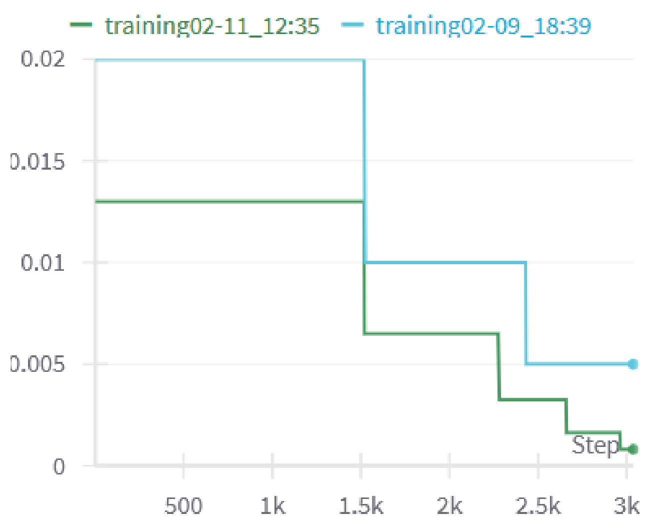


Fig. 7. Value of leanring rate for every bath ot the traning. Blue line stands for training the ResNet101 and the green one for TnceptionV3

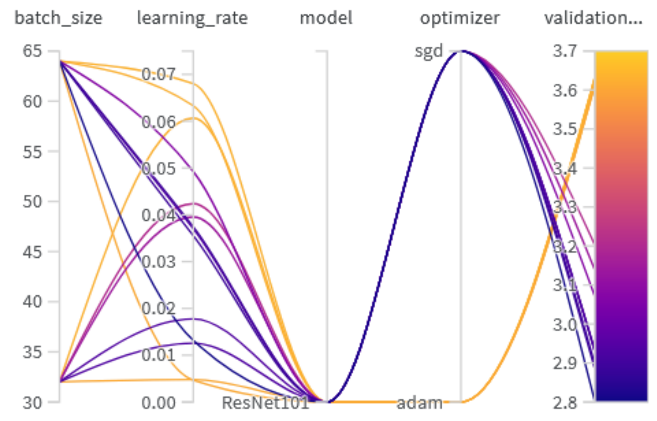


Fig. 6. Results of hyperparameters optimization using Weights and Biases sweeps for InceptionV3

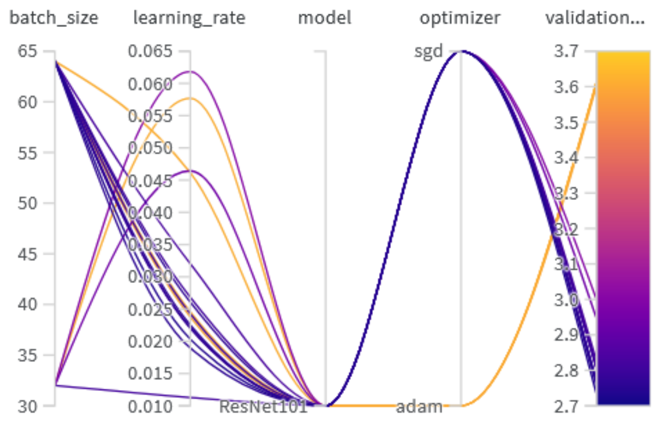


Fig. 8. Results of hyperparameters optimization using Weights and Biases sweeps for ResNet101

C. Regularisation Techniques

To prevent overfitting, two early stopping criteria are implemented: training halts after four consecutive epochs without improved validation accuracy, or when the training loss becomes lower than the validation loss by a specified threshold and the training accuracy exceeds the validation accuracy by another margin. Additionally, a learning rate scheduling technique is applied, dividing the learning rate by a defined constant after every three consecutive epochs without improvement (see figure 7). In order to control vanishing or exploding gradients, gradient magnitude was plotted every batch what is shown on figure 5.

D. Hyperparameters Tuning

For optimizing hyperparameters in the model, Weights and Biases Sweeps with Bayesian optimization were utilized like in Dobrovsky's work [1]. The Kaggle notebook created by Samuel Cortinhas [9] was very useful as tutorial how to set it up. The method systematically tests different combinations of settings to find the best ones. Bayesian optimization learns from each combination tried and uses that knowledge to guide the search towards more promising areas. By focusing on combinations that have shown good results in the past, Bayesian optimization efficiently refines the search space and identifies settings that improve the model's performance. As there are a lot of hyperparameters to optimize and computational resources were limited, only the initial learning rate, criterion and batch size were optimized. the results of the optimization can be seen on figure 6 and 8.

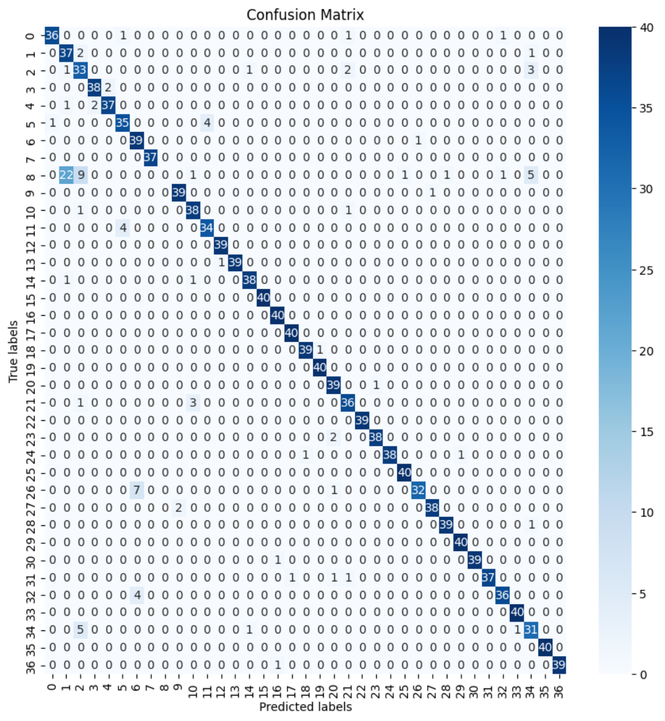


Fig. 9. Confusion Matrix for ResNet101

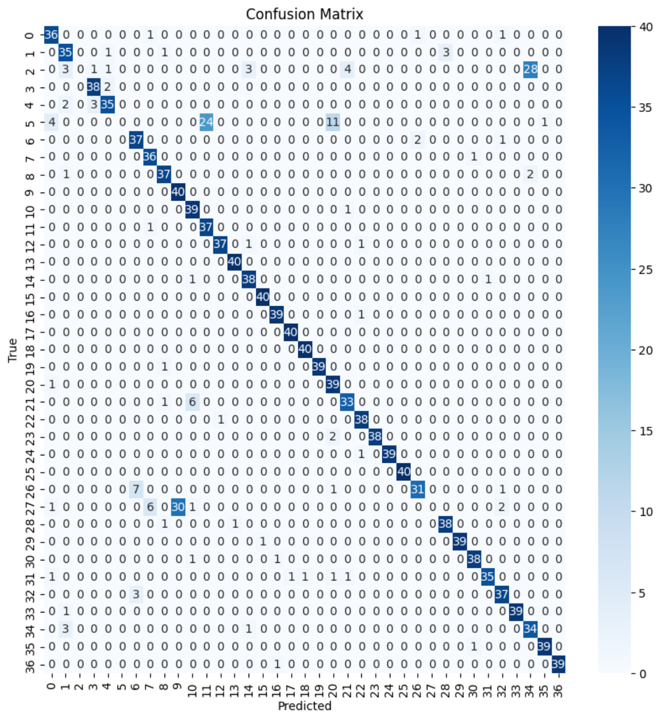


Fig. 10. Confusion Matrix for Inception V3

V. Evaluation

After the training was completed, both models were evaluated on the testing dataset, and accuracy scores were computed. The results are shown in the table I. The better model achieved 92,4% accuracy what was signifacantly better than model trained by Meet_Vora [3] on the same dataset and mentioned here VGG architectures applied for similar calssificaiton problem with smaller number of classes [6]. However results obtained by Aline Dobrovsky [1] were better on the more complicated problem with a use of the same architecture. Accuracies obtained on validation and testing datasets have very similar values indicating that modeles were not overfitting. For each of the models, the confusion matrix was created and plotted to visualise which classes have been mistaken the most often (see figure 9 and 10). It can be observed that for most of the classes model makes almost perfect predictions, so in this cases recall almost reaches 100%, but there are few classes where it always predicts different class (recall = 0%) what decreases the overall accuracy. Training of each of the models has stopped after reaching the limit of epochs which was 40, without triggering the early stopping.

Model	Test accuracy	Validation accuracy	Training accuracy
ResNet101	92,7%	91%	88,7%
InceptionV3	87%	86,4%	82,7%

TABLE I
Performance of the models

VI. Conclusions

This work was done for educational purposes, as a project for a university course. Its aim was to learn about the convolutional neural networks and how to apply them in practice. Therefore, the primary focus was on experimenting with different tools, debugging code, and conducting a literature review to understand various solutions to the problem. Many things in the project were done by the author for the first time. However, the results proved to be satisfactory and comparable to those reported in the literature, even outperforming some of the solutions. The ResNet101 model achieved better accuracy on the test and validation sets than InceptionV3. This is understandable since ResNet101 is a more complex model with over twice as many layers as InceptionV3, despite the fact that Inception modules are more complex than typical CNN layers used in ResNets. Perhaps even better accuracies could be achieved by tuning more hyperparameters through additional runs and by training for a higher number of epochs. Additionally, further exploration of data augmentation approaches and the application of more complex CNN architectures could be pursued in future work.

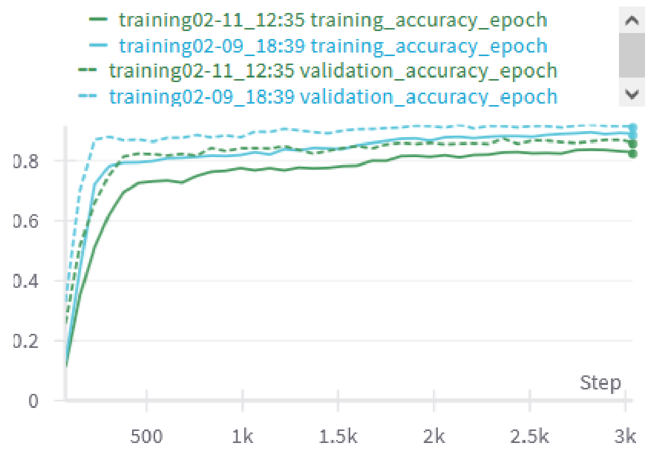


Fig. 11. Accuracies on the validation and training sets at the end of every training epoch for both models. Blue colour was used for ResNet 101 and green for Inception V3. Dotted lines correspond to validation accuracies and solid to training accuracies

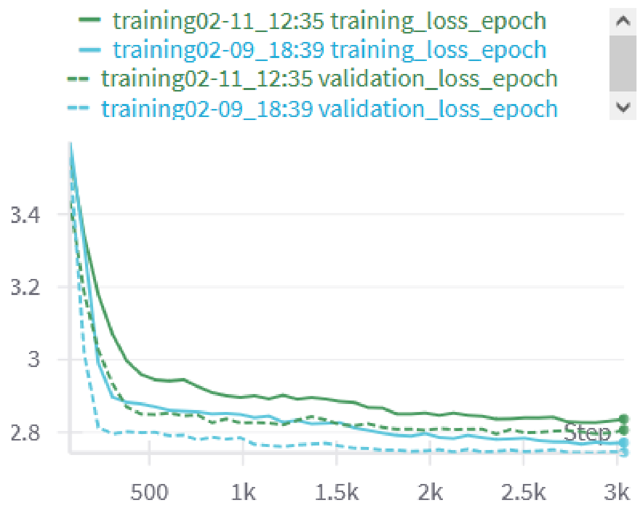


Fig. 12. Loss on the validation and training sets at the end of every training epoch for both models. Blue colour was used for ResNet 101 and green for Inception V3. Dotted lines correspond to validation loss and solid to training loss

References

[1] A. Dobrovsky, "Plant Disease Detection by Image Classification," https://moodle.aau.at/mod/folder/view.php?id=1138733, 2023, [Online]. Accessed: 2024-02-06.

[2] dr. Avicenna, "Cats and Dogs Breeds Classification Oxford Dataset," https://www.kaggle.com/datasets/zippyz/cats-and-dogs-breeds-classification-oxford-dataset, 2018, [Online]. Accessed: 2024-02-06.

[3] Meet_Vora, "Dog and Cat Breed Classifier - 88% accuracy," https://www.kaggle.com/code/meetvora090201/dog-and-cat-breed-classifier-88-accuracy, 2023, [Online]. Accessed: 2024-02-06.

[4] A. K. M. Ramaprasad Poojary, Roma Raina, "Effect of data-augmentation on fine-tuned CNN model performance," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 10, no. 1, pp. 84–92, March 2021.

[5] I. Dutta, "PetFinder: Data Augmentations Master Notebook," https://www.kaggle.com/code/ishandutta/petfinder-data-augmentations-master-notebook, 2021, [Online]. Accessed: 2024-02-06.

[6] Mahardi, I.-H. Wang, K.-C. Lee, and S.-L. Chang, "Images classification of dogs and cats using fine-tuned vgg models," in *2020 IEEE Eurasia Conference on IOT, Communication and Engineering (ECICE)*, 2020, pp. 230–233.

[7] P. Borwarnginn, K. Thongkanchorn, S. Kanchanapreechakorn, and W. Kusakunniran, "Breakthrough conventional based approach for dog breed classification using cnn with transfer learning," in *2019 11th International Conference on Information Technology and Electrical Engineering (ICITEE)*, 2019, pp. 1–5.

[8] K. Z. B. Liu, Y. Liu and. (2022) Image classification for dogs and cats. [Online]. Available: http://www-labs.iro.umontreal.ca/~liubang/files/DogCat_report.pdf

[9] S. Cortinhas, "Advanced WandB: Hyper-parameter tuning (sweeps)," https://www.kaggle.com/code/samuelcortinhas/advanced-wandb-hyper-parameter-tuning-sweeps, 2022, [Online]. Accessed: 2024-02-06.

VII. Replication Package

- The notebook with a data analysis is available at:
  - https://www.kaggle.com/code/jerzypawlik/data-analysis-cats-dogs-oxford
- The notebook used for doing the experiments can be accessed by:
  - https://www.kaggle.com/code/jerzypawlik/cats-dogs-project
- All the important graphs logs and results of sweep optimization with the hyperparameters configuration data can be found here:
  - https://wandb.ai/jerze/cats&dogs_ML&DL_project