Project in Deep Learning in Data Science

ABOU-NAJM Jennifer
HANGARD Kelyan
COHEN-DUMANI Joshua
OH William

June 29, 2023

Abstract

This project focuses on using transfer learning methods to solve an image classification problem. In particular, we use the ResNet34 and ResNet18 ConvNets to classify the images found in the Oxford-IIIT Pet Dataset. We also explore the effects of different parameters such as number of epochs and learning rates, as well as seeing how data augmentation and freezing some layers affects our results. We discovered that with transfer learning, it was possible to classify the image as a cat or a dog with an accuracy greater than 99%. Furthermore, we also managed to get an accuracy of more than 95% when trying to classify the breed of cats and dogs.

1 Introduction

In this project, we focused on exploring how transfer learning can be used to classify images of dogs and cats. We will be using the Oxford-III Pet Dataset, which includes 7349 labelled photos of 37 breeds of dogs and cats.

Transfer learning consist of adapting a pre-trained model to a different but related task (for example, within the realm of a classification problem). It is done by fine tuning the existing models using the data we want to classify. We usually fine-tune only the latter layers, since these are the most specific to the task at hand, whereas the deeper layers are more "generalist" (shown empirically in section 6.4!). This approach allows us to use relatively deep networks and achieve outstanding accuracy without much computing power, as the weights are already "pretty good" at (for example) classifying things. It is therefore much easier than training a full-blown deep learning model from scratch (which requires tremendous computing power).

In our approach, we have used both the ResNet34 and ResNet 18 models, which have 34 and 18 layers respectively. We started with ResNet34 but due to its heavy computational demands, we switched to ResNet18 for the experiments following the feedback. We will finetune the models to adapt them to the cats-and-dogs problem.

Our first problem will be binary identification (either "cat" or "dog"). For this problem, we expect to have an accuracy greater than 99%. Following this, we will further

finetune the model to make it proficient at recognising the breed of dogs and cats. We expect to have an accuracy greater than 95%. Our early attempts, in which we actually managed to reach these numbers, are detailed in section 5.

The experiments that we conducted following the feedback can be found in section 6. In those experiments, we explore the impact of data augmentation, and go into much more detail with the performances of our model as we increase the number of epochs and vary the amount of data. We also look into the impact of freezing one or multiple layers on the training and validation losses, as well as explore the impact of varying batch size during finetuning.

2 Related work

Deep learning methods have commonly employed image classification of dogs and cats as a case study. Because dogs and cats are reasonably easy to recognize, they are frequently used as examples for picture classification algorithms. In fact, there are numerous works that tackle the issue of classifying dogs and cats. Omkar M Parkhi [1] established a system for categorising images of 37 different dog and cat breeds with a 59 percent accuracy, and explored two classification approaches: a hierarchical one, in which a pet is first assigned to the cat or dog family and then to a breed, and a flat one, in which the breed is obtained directly. With a testing accuracy of 88.31 percent, S. Panigrahi [2] employed a deep learning model to identify photos of dogs and cats simply as "dog" or "cat". With 90.10 percent accuracy, Jajodia [3] employed a sequential CNN to make a similar basic differentiation, using 8000 samples of dogs and cats for training and 2000 samples for testing the model. To construct the new Resnet models, Reddy [4] , Deng [5], Lo [6] and Buddhavarapu [7] combined transfer learning approaches with Keras models.

3 Data

For our project, we used the Oxford-IIIT Pet Dataset. It is a 37-category pet dataset with around 200 pictures each class. The photographs feature a wide range of scale, animal position, and illumination. All photos contain a breed, head ROI, and pixel level tri-map segmentation ground truth annotation.

For the experiments in section 5, we were unable to use the full dataset for more than one or two epochs due to the massive amount of data. In order to see the effects of varying the number of epochs and data augmentation, we decided in our second set of experiments (section 6) to use a smaller amount of data (generally under 1000 images from the dataset). Although this will limit the performance of our model, we wanted to be able to conduct tests for many epochs.

4 Methods

Firstly, we downloaded the data Oxford-III Pet Dataset. We then divided the data set into a training data set of 5912 images and a testing dataset of 1478 images.

Then, we labelled the data as 'True' if the image corresponded to a cat and 'False' if the image corresponded to a dog. We achieved that using a labelling function that takes the first letter of the filename and determines if it's a cat or dog depending on if it is a lower or upper case letter (convention defined: uppercase: cat, lowercase: dog).

For the multi-breed classification problem, the convention was the same (lowercase for dog breeds and uppercase for cat breeds).

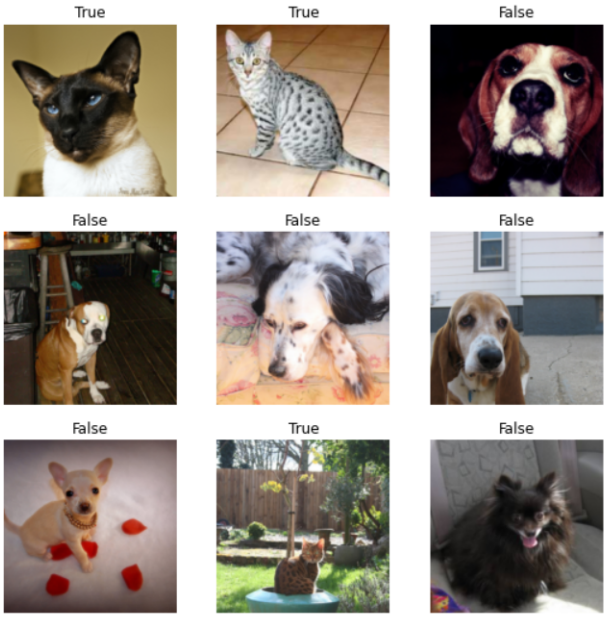


Figure 1: Binary labelling

Figure 2: Breed labelling

After that, we resized the image to 224 pixels by 224 to be able to use ResNet models. ResNet is a convolutional neural network that can be utilized as a state of the art image classification model; the ones we are using (ResNet34 and ResNet18) have 34 and 18

layers respectively.

In our first set of experiments, we used ResNet34 and adapted it to our dataset using the Adam optimizer and a learning rate of 0.001 (at first). We fine-tuned the system for one epoch by replacing the final layer of the pre-trained model to fit our data. We conducted some tests in section 5, aiming to understand the effect of various parameters. we had limited success, only managing to get data over 1 or 2 epochs, given that each epoch took several hours to compute.

In the second set of experiments (section 6), we used the ResNet18 model, which is less computationally demanding. This allowed us to conduct much better tests, and draw concrete conclusions.

5 Experiments: Full dataset and ResNet34

When using the full dataset and the 34 layer model, our run time was extremely long. For this reason, our goal was to achieve the desired accuracy using a mix of theoretical knowledge and what empirical data we could gather. In section 6, more complete data can be found with more concrete empirical evidence, in order to see the effects of different parameters in more detail.

5.1 Performance with ResNet34

5.1.1 Binary classification

We first ran the model with default parameters on the binary classification problem, using the full dataset and ResNet34. We found that even with a single epoch, we already had an accuracy of well over 99%, which shows the power of transfer learning. We were not able to run more epochs with the complete dataset, as this single epoch already took very long to compute. The results can be observed in table 1.

epoch	train loss	valid loss	error rate
0	0.035641	0.013674	0.004060

Table 1: Loss and accuracy of the model on the binary classification problem after fine-tuning

5.1.2 Breed classification

In order to achieve our desired accuracy ($\geq 95\%$), we decided to find the loss for many learning rates in order to find a good one using an in-built function in fastai. The results can be observed on figure 3.

We then fine-tuned our model with the learning rate we found (0.00062). The results are detailed in table 2. From these, We can see that even though we only were able to run 2 epochs, we were able to very quickly reach an accuracy of 95,57%. This, again, shows how powerful the ResNet models are when you use large amounts of data and have access to a lot of computing power.

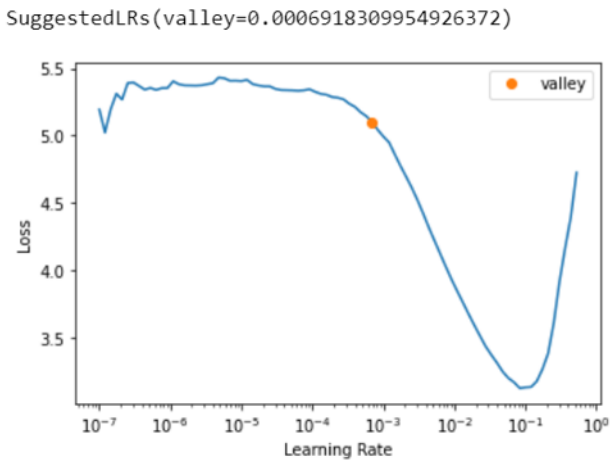


Figure 3: Graph to determine a good learning rate

epoch	train loss	valid loss	error rate	time
0	0.549484	0.439747	0.127199	28:35
1	0.316820	0.261843	0.044321	3:02.14

Table 2: Loss and accuracy of the model on the multi-breed classification problem after finetuning

5.2 Varying the number of epochs

We tested different numbers of epochs to see the effect on the network performances. We know that one epoch is when an entire dataset is passed one time forward and backward through the neural network. In theory, as we increase the number of epochs, the model gradually moves from being underfit, to overfit, and with an optimal amount of epochs, it will maximize the accuracy of the learner.

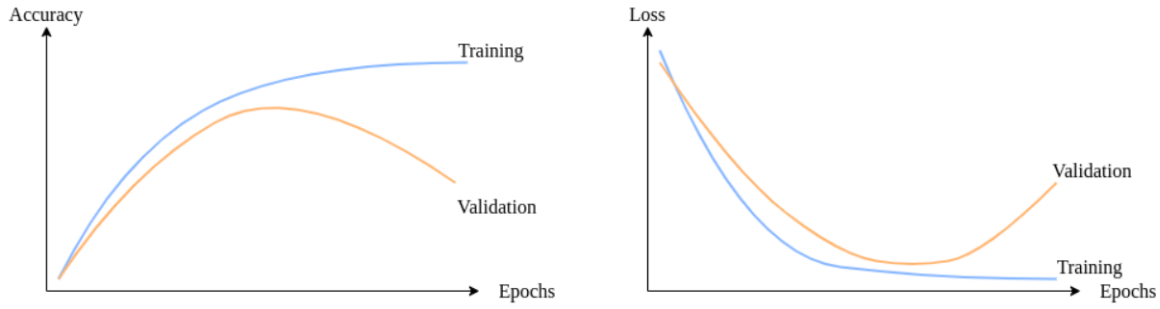


Figure 4: Theoretical effect of number of epochs on the network performances

In order to see this effect through experimentation, we used the learner.fit_one_cycle() function provided in vision module to estimate the effect of changing amount of epochs *relatively* quickly, since it uses a very large learning rate (not at all optimal but perhaps faster convergence). We tested this for the binary problem since it was simpler and took less computational effort.

As we can see in figure 5, the training loss and valid loss are both decreasing as the number of epochs is increasing. It follows our predictions from the figure 4, we also notice

```
classifier = vision_learner(labeled_data, resnet34, metrics=error_rate)
#classifier.fine_tune(1)
#classifier.fit_one_cycle(1)
classifier.fit_one_cycle(2)
```

epoch	train_loss	valid_loss	error_rate	time
0	0.145603	0.006618	0.003383	35:02
1	0.054982	0.004419	0.002706	34:42

Figure 5: Network performances after one or two epochs

that the training loss is decreasing faster than the validation loss. Furthermore, we see that the error rate is decreasing as we increase the number of epochs, which also follow our previous interpretations that the network is fitting more the training data with the number of epochs. However, we can only give limited credibility to these results, as we only have 2 epochs.

6 Experiments: ResNet 18 and partial dataset

In this section we changed the model used (we used ResNet18 instead of ResNet34), and lowered the amount of images we would be using for training and validation. Although this lowered our accuracy in the end, we were able to conduct real tests and draw empirical conclusions on the effect of various parameters.

6.1 Varying the number of epochs

Here, to investigate the impacts of varying the number of epochs, we conducted other tests using RestNet18 and a partial data set (500 images first, and then 100 images).

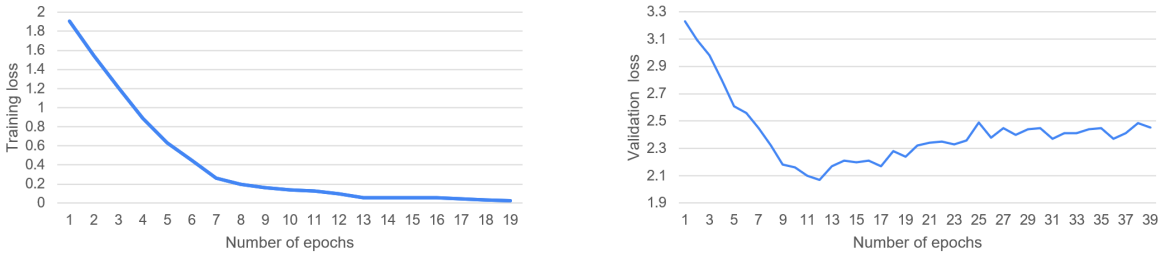
We noticed that when the number of epochs increases, the training loss also decreases since we pass the training data through the classifier multiple times and the model becomes increasingly good at recognising the training data. We can observe this on figure 6a.

For the validation loss, as seen on figure 6b, we can see that it decreases in the beginning, which we can expect since we are fine-tuning our model to be more effective for this dataset by using a bigger number of epochs. However, at some point the validation loss will start increasing again, since we will start to overfit the model to the training data. That comes from the fact that the model is now overfit to the training data since we used a very big number of epochs on a reduced number of images (only a hundred images in the graph to exaggerate the problem that might arise).

6.2 Exploring the effects of data augmentation

In order to explore the effects of data augmentation, we decided to use 100 images. This allows us to really visualise the effects of this technique, as we will be using few images, thus exaggerating the effect we might have. The results can be observed on figure 7.

The results clearly show that even though we have the same amount of raw data as



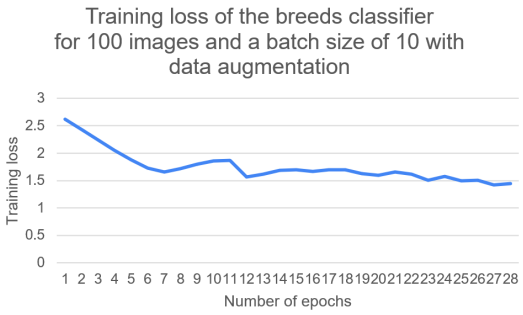
(a) Training loss (500 images)

(b) Validation loss (100 images)

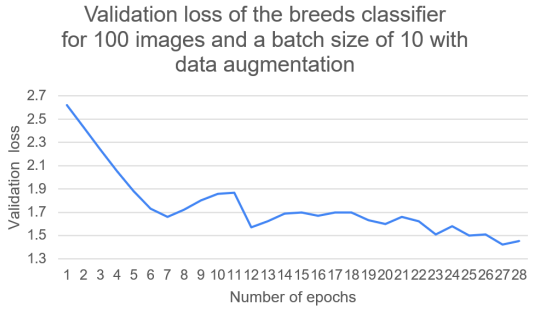
Figure 6: Loss for the multi-class problem, with a batch size of 10

in section 6.1, we do not see any overfitting! Indeed, the validation loss continues to drop even after 39 epochs (figure 7c). Without data augmentation, we started to overfit the model after only a dozen epochs (as seen on figure 6b). This gives us much better hope of generalization, as we are able to make the loss go much lower. For an identical number of images, we were able to go from around 2 (local minimum without data augmentation, likely to be variable depending on data) to under 1,3 (with a slow convergence showing more resilience and thus better generalization).

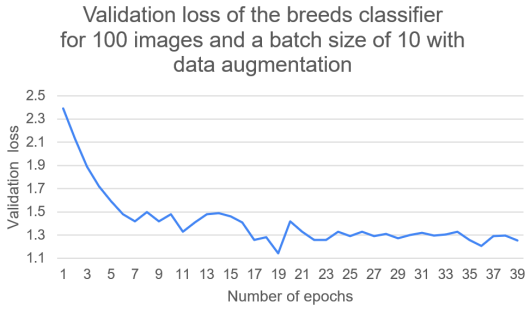
We also notice that data augmentation has a negative effect on the training loss. We saw in the figure 6a that the training loss start from 2 and quickly tends to 0 within a relatively low number of epochs without data augmentation. In the figure 7a, we see that the training loss start from 2.5 and tends to 1.5 with data augmentation. We could explain this by the overfitting of the model without data augmentation which leads to a zero training loss and a generalisation of the model with the data augmentation which leads to a residual training loss. In the end, it doesn't matter that much considering that the most important performance parameter is the validation loss, but it was interesting to note nonetheless.



(a) Training Loss



(b) Validation Loss run 1



(c) Validation Loss run 2

Figure 7: Effects of data augmentation on training and validation losses

6.3 Varying batch size for the breed classification experience

The batch size is a hyper-parameter that defines the number of samples to work through before updating the parameters of our model. In order to see the impact of this parameter, we decided to do multiple runs while varying only the batch size, keeping other parameters equal. Here we see that a lower batch size seems to lead to a quicker convergence and a lower loss function for validation and training. This experience was realised with a sample of 500 images. We also a notice a speed difference in function of the batch size for the calculation of an epoch but it is not that significant. For a batch size of 15 it takes 1 minute 55 seconds to calculate an epoch. For a batch size of 25 it takes 1 minute and 52 seconds. For a batch size of 35 it takes 1 minute 47 seconds. For a batch size of 45 it takes 1 minute 40 For a batch size of 65 it takes 1 minute 36 seconds.

From figures 8 and 9, we can that higher batch size leads to lower asymptotic test accuracy. Using a batch size of around 15 seems then to be a good choice to minimise loss.

Figure 8: Batch Size Variation - Training loss (y-axis, x-axis = epochs)

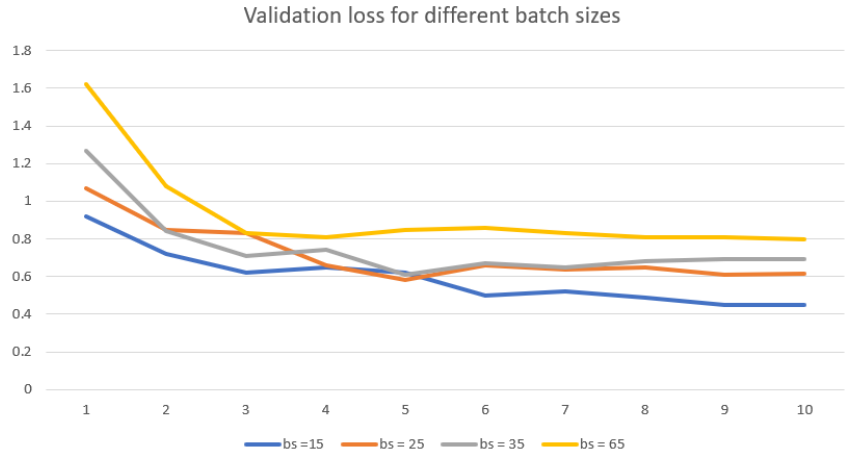


Figure 9: Batch Size Variation - Validation Loss (y-axis, x-axis = epochs)

6.4 Freezing different parts of the model for the breed classification experience

When the weights of our pretrained model are unfrozen we see that there is bigger variation in the validation loss which can be explained by the overfitting of the training set as we modify the first layers which are already well trained in the basic patterns of image recognition.

The Resnet18 model has 3 layer groups, 18 layers and 11,722,816 total parameters. When the model is unfrozen, all parameters in all 3 groups are modifiable. The model is "partially frozen" when the first layer group is frozen, and the rest modifiable. In this case, there are 11,041,664 trainable parameters. Finally, when the model is frozen, then the 2 first layer groups are frozen and the last one is modifiable. In this case there are 555,904 modifiable parameters.

We can see that the partially frozen model and frozen model (in which only the last layer is modified) seem to converge more quickly. However, we also see that the loss validation of the partially frozen model values fluctuate more than the frozen model before

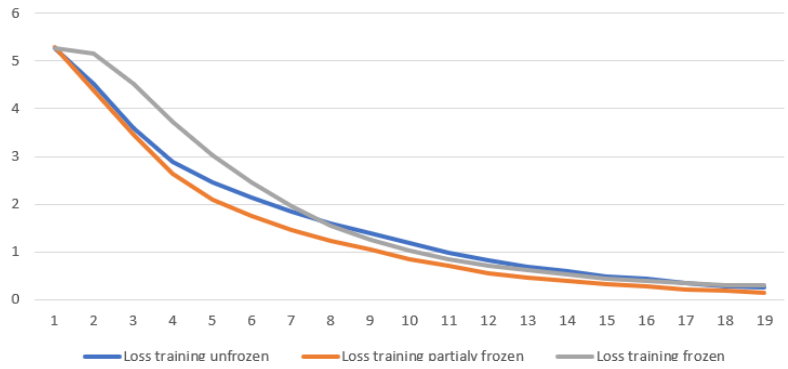


Figure 10: Effect of freezing parameters on the training loss (y-axis)

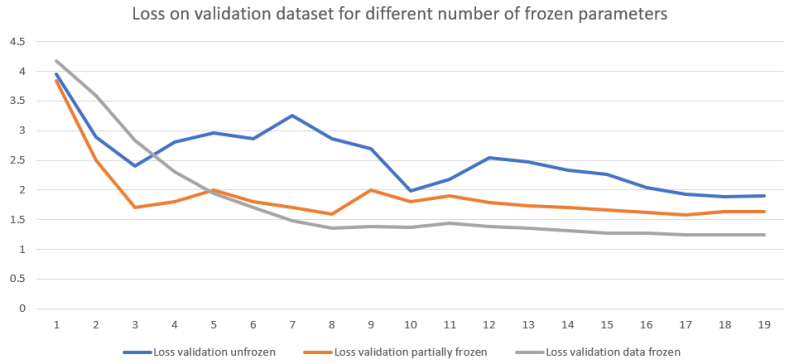


Figure 11: Effect of freezing parameters on the validation loss (y-axis)

stabilising.

The experience was realised with a sample of 250 images to reduce the computational time. The frozen model had a computational time of 42 seconds for each epoch. Which is lower than for the partially frozen model which had 50 seconds and the unfrozen model which had 57 seconds. This was expected as the more layers of the model are frozen the less parameters are adapted leading to less computing time.

7 Conclusion

In conclusion, this project enabled us to get a better understanding of transfer learning. Through implementation of the image classifier, we learned how to take advantage of the vision modules from fastai. From our results, we saw how powerful transfer learning could be (we achieved outstanding results with only 1 or 2 epochs when using the full dataset and ResNet34!).We saw how different parameters interact with the performance of our model, as well as what trends we can observe in the data with a transfer learning approach (underfitting and overfitting dynamics as epochs increase for example). We conducted data augmentation and saw how freezing certain layers of our model affects performance (showing, again, how powerful these models are as is!).

We also learned the different computational demands required when using different models, and adapting this to our computational budget.

In a future cycle, we could try to find optimal values for these parameters in order

to maximize the accuracy in our given problems. Given that with minimal optimization, we managed to get excellent results, we could build a very powerful model with more experimentation. It could also be interesting to try and use different methods of data augmentation, and perhaps changing the loss function.

References

[1] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. V. Jawahar, *"Cats and dogs"*, pp. 3498–3505. in IEEE Conf. on Computer Vision and Pattern Recognition, Providence, RI, 2012.

[2] S. Panigrahi, A. Nanda, and T. Swarnkar, *"Deep learning approach for image classification"*, pp. 511–516. in 2nd Int. Conf. on Data Science and Business Analytics (ICDSBA), Changsha, 2018.

[3] T. Jajodia and P. Garg, *"Image classification - cat and dog images"*, pp. 570–572. International Research Journal of Engineering and Technology, vol. 6, no. 12, 2019.

[4] A. Reddy and D. S. Juliet, *"â€œTransfer learning with ResNet-50 for malaria cell-image classification"*, pp. 0945–0949. in 2019 Int. Conf. on Communication and Signal Processing (ICCSP), Chennai, India, 2019.

[5] G. Deng, Y. Zhao, L. Zhang, Z. Li, and Y. Liu, *"Image classification and detection of cigarette combustion cone based on Inception Resnet v2"*, pp. 395–399. n 2020 5th Int. Conf. on Computer and Communication Systems (ICCCS), Shanghai, China, 2020.

[6] W. W. Lo, X. Yang, and Y. Wang, *"An Xception convolutional neural network for malware classification with transfer learning"*, pp. 1–5. in 2019 10th IFIP Int. Conf. on New Technologies, Mobility and Security (NTMS), Canary Islands, Spain, 2019.

[7] V. G. Buddhavarapu and A. A. J. J, *"An experimental study on classification of thyroid histopathology images using transfer learning"*, pp. 1–9. Pattern Recognition Letters, vol. 140, 2020.