COP528 –Ai and machine learning

I. Introduction

Machine learning (ML) has been emerging as a force that can transform data across a plethora of applications, both shaping and redefining industries and the problem-solving approaches they may have. From finance and health care to autonomous vehicles and natural language, ML has a vast and diverse application potential.

Here I will include two tasks, task one will demonstrate an understanding on how machine learning approaches are devolved to satisfy requirements from various applications and task two will be another report based on the data set given to me.

II. Broad spectrum of machine learning applications Task 1

Introduction:

There's an abundance and variety of applications that machine learning can offer. With many different domains be benefitting from machine learning and the boost in technology I'll go over some.

1) Healthcare

In healthcare, Machine learning is integrated into many different areas such as treatment plans, diagnosis of diseases and even drug discovery. According to Esteva et al. 2017[1], deep neural networks demonstrated a huge potential in skin cancer detection and classification within the dermatology sector. Huge data sets of dermatologist images were used, and the machine learning algorithms were able to identify and classify the skin lesions with great accuracy. The results of the neural networks rivalled that of experienced dermatologists. It not only improved the patient outcomes by increasing the rate of early detection but also allowed for more efficiency, cutting of capital cost and the strain that is quite common on the workforce within health care.

2) Finance

Machine learning has become revolutionary tool within the finance sector such as risk assessment, algorithmic trading, and fraud detection. In the study, Tsai et al. (2009) [2], it demonstrated that the use of SVM or support vector machines for credit scoring and the analyzation of vast transactional data via the use of models that would detect real-time fraud followed by inform lending decisions all while minimizing risks. Furthermore, Zohren and Biloken (2018) [3] showed cased that within algorithmic trading, machine learning can help leverage historical market data and analyses automated trading decisions. However, some problems were raised such as model interpretability. When looking at all these benefits it's hard to say that machine learning is not crucial to the overall performance of a dynamic market such as trading.

3) Autonomous Vehicles

Lastly, Machine learning can be appleid within the vehicle industry as the new wave and crave is the research into self driving and automated cars. Bojarski et al. (2016) [4} demonstrated that end-to-end learning approaches for self-driving cars, the leveraging of deep neural networks and driving decisions were possible within the automated industry. Companies such as tesla are using machine learning algorithms to improve and prolong the use of self driving cars, autoamted repsonses etc. However, until the use of CNN,s for image recognition deep reinforcement, deep reinfrocement and superv ised laernign for mapping drastically improves self dirivng cars will not be massively used on a prolonged time preiod due to safety concerns. This is evident due to companeis such as tesla limiting the time period In which the car can self drive.

Strengths & weaknesses of machine Learning methods

Below I will discuss the strengths and weaknesses of machine learning methods highlighted in the studies:

Hastie, T., Tibshirani, R.., & Friedman, JK. (2009). The elements of statistical learning: Data mining, inference, and prediction. Springer. [5]

Bishop, C. M. (2006). Pattern recognition and Machine Learning. Springer. [6]

Superviused learning:
Supervised learning is well suited for task such as data being labelled, high accuracy predictions and robust generalization. On the other hand, it's quite dependent on the labelled datasets, has limited applicability to said tasks which do not operate under clear named labels and is quite susceptible to over fitting.

Unsupervised Learning:
Where supervised learning lacks, unsupervised improves on. This is shown in its improved generalization, meaning it doesn't rely on labelled datasets and can operate on a multitude of datasets. The ability to reduce overfitting, by using dimensionality reduction, clustering, data augmentation etc. It helps discover hidden patterns using these methods. However, it's quite dependent on the quality of data and has challenges when evaluating the performance of a model.

Ensemble methods:
This method is a powerful tool that will leverage the collection of multiple models to increase and enhance the prediction performance. It improves generalisation by combining a diverse number of predictions, consequently causing the risk of overfitting to reduce and provide a more robust and reliable set of results on the unseen data. However, with these positives come negatives, as the ensemble method is quite complex, thus leading to an increased cost due to both implementing and trying to interpretate multiple models. Also, if the models are overly complex or highly correlated, the risk of overfitting will still occur. Despite these cons, this methods value cannot be underestimated as it can refine predictions via the use of varied perspectives and multiple model's insights.

Transfer Learning:
This method is a powerful tool as it improves performance on a target task by leveraging knowledge from a related source domain, such as that which is limited by labelled data. It will additionally reduce the data requirements and accelerate the training by applying a pre-trained model with general features from with abundant amount of data. The effectiveness, however, is dependent on the similarity of the source and target domains, in which there is a risk of overfitting if the transfer is not regularized carefully. Though issues may arise, transfer learning remains one of the most valuable tools for efficiently adapting models to new tasks and domains, even if the labelled data is costly to obtain or insufficient.

Challenges and Ethical concerns in ML Algorithms

Challenges:
Machine learning algorithms have an issue concerning data biases and quality as biases may be president when training data which can result is skewed or often ML models that are not actually a good

representation. To address these biases and discriminations within the model, one must put time and effect to ensure the model has a high interpretability and that to mitigate this risk, one must also meticulously handle the data and algorithm design. This is true according to Obermeyer, Z., & Emanual, E.J. (2016) [7] in where they found that these challenges occur when integrating ML into sectors such as the ones I discussed earlier.

Ethical Concerns:

Machine learning may be useful and crucial to the rise technology integrated into work and life in general, however it is not without its ethical concerns. Various ethical aspects are raised including privacy, fairness transparency and the handling of sensitive information. This is prevalent in the medical field as machine learning algorithms are used to investigate an individual's DNA and genes to recommend treatment, the risk of other diseases etc. This is shown in the study Esteva et al. (2017) [1] previously spoken about, in which they could get someone's Family DNA and accurately predict the risk of diseases. But where is the privacy of the individuals not involved in this treatment? Lo Piano, S., (2020) [8] found that there is an importance in fairness and transparency within the machine learning applications and that ethical guidelines and rules must be available to the wider public and updated as the technology continuous to grow and be integrated. In a world where Ai development and deployment is rapidly increasing, both the safety/privacy concerns and ethical guidelines must be addressed.

III. Task 2: Applying Machine learning Appraoch(es) in an Application.

Introduction:

The task given involves the classification of images containing various objects into their corresponding classes. The report presents the development and evaluation of three distinct machine learning models for image classification: A custom model, ResNet, and VGG16. By comparing and evaluating these approaches, I can then aim to identify the most effective and efficient strategy for image classification and gain insights into the accuracy performance of the different models. Via this, I could see if the models used can contribute to the advancement of image classification techniques and their practical applications within various domains.

Preliminary Data Analysis:

1. *Initially, I converted the image tensors into `float32` data type. This was done by using the `type.cast()`, which casts the data type to a specified data type, in my case `float32`. This was done to convert the images into float form, which is quite common in pre-processing steps, and allow for the numerical operations to occur which is then required for the next step of normalization.*
2. *The pixel values of the images are then normalized to be within a range of [0-1] and this is achieved by then dividing each pixel value by 255. This was done as I found it to be the maximum pixel value for images with 8-bit colour depth. Normalisaiton was used as it ensures the input features have the same scaling, which then can improve the stability of training and the eventual convergence of neural network models.*

Visualising the data

Once I did this, I could then visualise the data by providing a code snippet for each data set that would import a random set of pictures. I initially loaded in the dataset from a directory specified by either variable `train_data` or `val_data. It's crucial to understand that val and test will be interchangeable as val stands for validation . During this process, the data set is shuffled to ensure randomness in the order of images during both training and testing, as shown by the parameter `shuffle = True`. This was done so that wouldn't keep getting the same images and I knew that the function worked in retrieving a random collection of images. As the images were organized into batch es of 32, each then resized to standardized dimension of 256 by 256 pixels. This aims to provide a representative sample of images from both training and testing to both inspect and validate. The code shows that I can plot a total of 9 images on and by using `plt.subplot()` it enabled me to visualise the images on an individual subplot and then, with some preliminary conversion using `plt.imshow()` and `uint8` I can ensure that the data type is compatible with matplotlib's rendering. Further coding was ensured so that the class name is presented as a title and that the axis are disabled for each subplot to enhance visual clarity.

Another visualization tool was used to dictionaries the `class _heights` and `class_widths` to store the height s and widths of sampled images for each class. This was done to both training and testing datasets to understand and visualise the distribution of image sizes within each class. Once all images and their respected sizes were collected , a box plot and whisker plots were generated so that the distributions of heights and widths for each class was visualised. Using these tools, I can clearly see that an assumption of class separability can be made, which I can assume that the classes within my dataset are separable, meaning there are distinct patterns or boundaries that would differentiate between one another. Furthermore, I can assume that due to the code working quite well my model with unseen data, in other words, there will be patterns learnt from both training and testing data that will be applicable to any unseen data. However, it's to be reminded that when downloaded I unzipped and ran some measures to ensure that all data is to be seen. Lastly, another assumption can be the accuracy of labelling. I can assume this as when looking at how my code to visualise accurately displayed the class labels, which corresponded to the files. I can assume that the label accuracy is going to be crucial to a reliable model.

Methods:

Here I will discuss the three models I used and their respective methods such as Transfer learning(TL), Convolutional Neural Network etc. For each model I changed the epochs to see whether the increase or decrease in epoch would affect the accuracy for both training and testing datasets.

My own model(CNN):

Methodology & Theoretical understanding:

I built a CNN model from scratch that involves designing a neural network architecture without the leveraging of a pre-trained weights or feature e extractors. In other words, not using a pre -trained model that already has learnt feature representation from a large dataset. By beginning with a blank canvas and then defining my own architecture from the ground up.

What this allows me to do is have full control of the architecture, including the layer numbers, type of layers e.g. convolutional, pooling, fully connected etc.), and the functions. All this allows the model to fully learn its own feature representation directly from the input data during the training process.

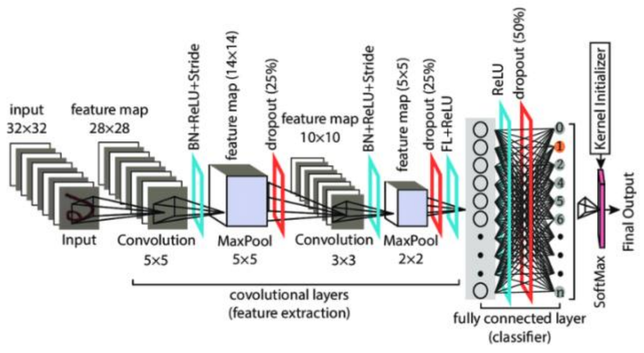

Figure 1- Convolutional Neural Network (CNN) Model [9]

Other advantages can be seen from:

- *Flexible and adaptability:*

CNN can be adaptive and customized for various task and datasets. I can modify the depth, width, and structure of CNNs to suit specific requirements.

- *Hierarchical feature learning:*

There is an automatic learn hierarchical representation of features directly form the input. This allows the convolutional layers to detect low-level features such as texture edges etc. Ultimately enabling CNNs to extract meaningful information effectively from both simple and complex datasets.

- *Parameter sharing:*

CNNs will exploit parameter sharing, where the filters/weights of the same set are applied across a different spatial location within the input. Thus. Sharing reduces the parameters amount within the network allowing for a reduced risk of overfitting when dealing with large images.

- *Robust to Variations:*

These types of models demonstrate a robustness to variations. Meaning that the variations in input data, be it change in light conditions, unseen data, background clutter etc., don't affect the model as much allowing the CNNs to both maintain a high performance and predictive accuracy across a diverse condition. Thus, making them very suitable for , in my case, Image classification but also computer visions tasks, object detection. Segmentation etc. Zhao, X., Sun, P., Xu, Z., Min, H. and Yu, H., (2020) [10] found that Object detection algorithms suitable for processing both camera data and LIDAR were mainly CNN's as they had an ability to learn complex features from a multimodal input. This was even when they were adapting existing algorithms or developing novel approaches tailored to multimodal nature of the input data.

As you can see in my code, I was able to define a sequential model comprising of convolutional layers, flattening layers, fully connected layers, and max-pooling layers. Each layer is made in mind for a specific set of parameters, functions etc.

i.e., The use of `ImageDataGenerator' for data augmentation. This data augmentation allows the train model to get more generalized and perform well of different sets of data. As the input data can come in varying of sizes it needs to be normalized to both a fixed format and size before the batches of data are used together.

Transfer learning: ResNet & VGG16: Methodology & Theoretical understanding:

For the use of Transfer Learning, I used both the VGG16 and ResNet models to have pre-trained model that were trained on large data sets, such as ImageNet. Both these models are known to have learned rich representations of features from a diverse range of images. Thus, allowing for faster convergence, due to being pre trained on large amount of labelled data and resources, better generalisation to new data and a potentially higher performance for my task. This is due to the fact the pre-trained models have more than likely learned to capture relevant visual patterns from a vast pool of images.

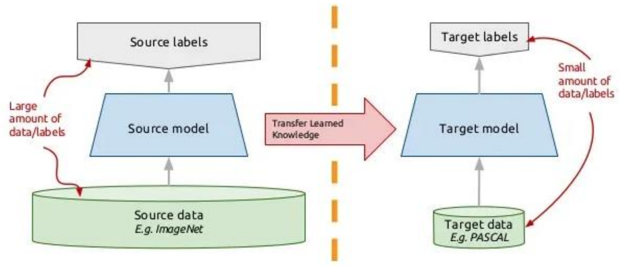

Figure 2- The Process of Transfer Learning [11]

Now that I've spoken on why I chose those pre-trained models, allow me to speak on the advantages:

- *Feature extraction:*

Transfer learning allows me to leverage the pre-trained models as feature extractors, so instead of training the netire model from scratch , I can remove the fully connected layers or top layers for both the VGG16 and ResNet models, but also retain their convolutional base. I need this convolutional base as it contains the convolutional layers that the model has previously learned to extract hierarchical features from other images. It will start with edges and textures , known as low- level features and then move to other features like object shapes and parts. These other parts are known as higher- level semantic features.

- *Fine tuning:*

In my case, I can add custom layers to the pre-existing layers on the pre-trained model. This is called fine tuning. It involves the use of updating the layer weights during training to better fit the new task, image classification, and allows it to adapt to learn a representation better suited to my dataset. This was all done to potentially increase and improve performance, accuracy, and generalisation.

- *Task-specific learning:*

I can add custom layers such as global pooling layers, dense(fully connected) layers, drop out regularisation(prevents overfitting by promoting redundancy in learned representation) and activation functions. This is done to enable the model to acclimate to the new task specific patterns and dataset. The understanding behind it is that while the pre-trained model is capturing generic features, the custom layers will fine tune the model to recognise unique and specific patterns related to my task. Which, in this case, is object classification in images.

The primary focus and motivation of transfer learning using both VGG16 and ResNet is to leverage the knowledge encoded in the pre-trained models and by doing so, I can benefit from features learn by *these models which will both accelerate the model development process due to reducing the amount of labeled data required for training and also potentially achieving a better performance compared to that of training a CNN model from scratch.*

Experiments Findings

This section will provide a comprehensive analysis of the machine learning experiments conducted to address the image classification task given. Through various architectural choices, training strategies, and tedious experimental, I've gained valuable insight into both the performance and behavior of my models. I will highlight the best-performing models and key findings derived from the use of learning curves and statistical analyses.

It's important to note that I involved 2 training strategies for all models, thus producing 6 codes and outputs. The primary focus was changing the number of epochs to access their influence on the model's overfitting and performance. The number of epochs would represent the number of complete passes through an entire data set during model training. This goes for both testing and training datasets.

This allowed me to further investigate trends, and patterns in performance metrics as the number of epochs increased. Allowing me to gain crucial insight into a model's learning behavior and how good its ability to generalise unseen data is. Furthermore, by increasing the epoch levels from 5 to 10 for each model, it allowed me to also investigate the model's ability to handle overfitting. This overfitting may occur when a model will learn to memorise the training data rather than capturing patterns that generalise to new data. This is more prevalent in CNN than transfer models but can happen in both. So, by introducing techniques such as regularisation dropout I can identify the point at which overfitting may occur as the epoch levels vary.

CNN model from scratch

CNN architecture:

I created two CNN models from scratch that used a basic CNN architecture that consists of convolutional layers followed by max-pooling layers for feature extraction and spatial down sampling. The convolutional layers apply a learnable filter to the input image, extracting various features such as textures and patterns. By doing this the model can learn increasingly complex patterns in deeper layers via the capturing of spatial hierarchies of features. Whereas max pooling reduces the networks computational complexity but retaining the most important features found. Thus, by selecting the maximum value within each pooling window, I can effectively discard information that is redundant and retain features that are needed.

So that overfitting doesn't occur, I introduced a dropout regularisation rate of 0.5 . This would randomly drop a fraction of neurons during training, preventing the model form relying too heavily on specific features and then therefore including and promoting a more robust type of learning.

Model Training and Evaluation:

When comparing both CNN models which varying epochs of 5 and 10. The model that trained for 5 epochs showed a higher validation accuracy compared to the model trained for 10. However, it wasn't by much as the epoch 5 model had a testing/validation accuracy of 68.20% whereas the 10-epoch model has a testing accuracy of 63.77% This could be due to several factors such as saturation of learning where after a certain number of epochs the model may have learnt everything it can from the testing data set. Thus, causing it to reach a plateau and any subsequent increase in further training may not yield significant improvements.

Another explanation could be that overfitting started to occur as after introducing a 6th epoch level, the model training accuracy did not increase and after the 7th epoch the `val_loss` started to increase indicating the performance started to dip. Suggesting the model predictions started to overfit due to deviating from true labels. It's safe to say that at the 5th epoch I saw the best overall ability to generalise as both the `val_loss` and testing accuracy was at its highest.

Where the 10 epoch CNN model lacks, it makes up for it in the training data accuracy. With an accuracy of 99.29%, it showed that the model flourished when looking at the training data set. The 5-epoch model didn't do bad as well as it had a training accuracy of 91.94%.

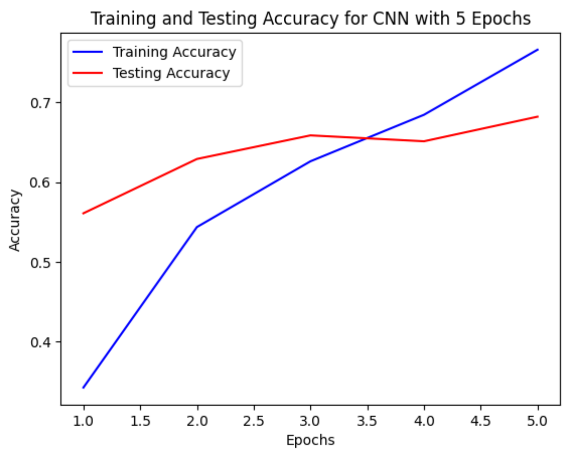

Figure 3- Training and testing accuracy for CNN model with 5 epochs.

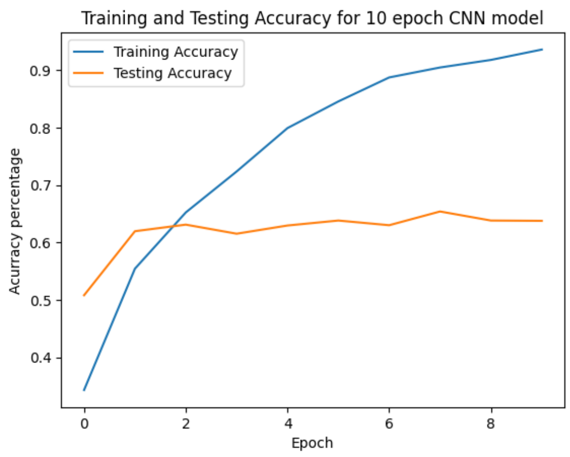

Figure 4- Training and testing accuracy for CNN model with 10 epochs.

Transfer Learning via VGG16

VGG16 architecture:

Both models would utilise the VGG16 architecture as a pretrained model on ImageNet, excluding the top classification layer in which I built myself. The base model's architecture is CNN model that was introduced by the uneivsity of oxford. Itys known for its simplicity and effectives in image classification. Just like a typical CNN/ Model I created it has max pooler layers that follow a stack of convolutional layers. When using this model as a base for transfer learning, the weights of the convolutional layers are initialised with the weights of the training on the ImageNet dataset. This dataset contains millions of labeled images that spans thousands of classes therefore giving me a rich representation learned by the VGG16. Thus, allow *me to apply this to my task. I removed the top classification layers of the pre-trained model, which were designed for ImageNet's classification and replaced them with top layers that a needed for my task in. These would be the same top layers applied in my CNN model.*

Model Training and Evaluation:

Both models showed consistent improvement over the base CNN model ,at both epochs of 5 and 10, reaching 90.24% and 91.75% respectively. They also showed very good testing/validation accuracy at 87.65% and 88.71% respectively. Both results were then followed with a consistent decrease in training and validation loss as the epochs increased demonstrating that the models were learning and improve the predictive capabilities needed for image classification

When analysing this data, I can conclude that both models showed amazing generalisation performance, with testing accuracy close to training accuracy. Suggesting, that the models were not significantly overfitting to the training data. There was a slight improvement as the epochs increase, not anything significant however it demonstrated even as the epochs increased, there was no sign of overfitting. They both always provided good convergence with both training and validation loss decreasing steadily and consistent throughout the training. Indicating that the model would effectively learn from the datasets while avoiding overfitting.

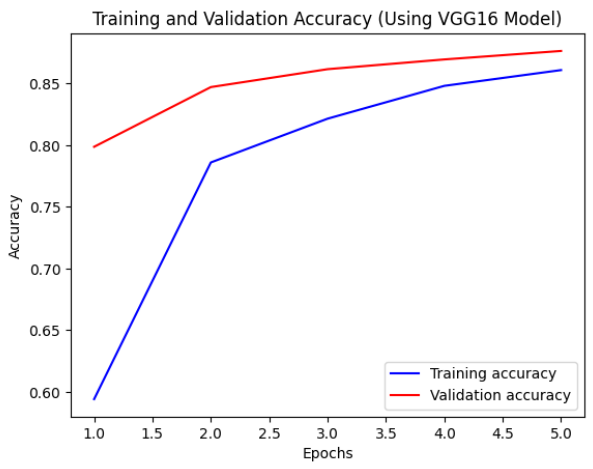

Figure 5- Training and Testing accuracy for VGG16 Transfer Learning 5 Epoch.

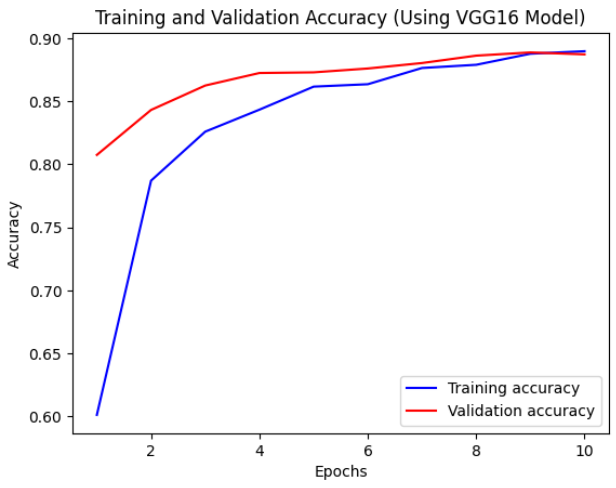

Figure 6- Training and Testing accuracy for VGG16 Transfer Learning 10 Epoch.

Transfer Learning via ResNet

ResNet Architecture:

Unlike the other models I ran, I decided to have 3 runs on ResNet at varying Epochs of 5 , 9, and 10. This was done to accommodate for a massive drop in validation and training accuracy that will be explained later.

All the models used a ResNet50 Architecture pre-trained on ImageNet just like the VGG16 one with the same excluded top classification layers. This allowed me, just like the VGG16, to leverage pre-learned feature representations from the large and diverse dataset. However, ResNet50 is known to have a deeper and more complex architecture than that of VGG16, thus allowing it to capture a varied number of intricate details and features.

Similarly to the VGG16 models, I used a basic. ResNet architecture, followed by a global average pooling layer and dense layers for classification. The same Dropout regularisation was applied to mitigate overfitting just like all the models prior.

Model Training and Evaluation

Unlike the other models, I ran 3 models with an epoch level of 5, 9, and 10. Model trained for 5 epochs came out with a training and testing accuracy of 46.26% and 44% respectively. Whereas the epochs for 10 achieved a training and testing accuracy of 12.24% and 12.54% respectively. Once's I saw that the epoch for 10 had a massive decrease after 9. At 9 epochs on the 10 epoch model the training and testing came out to be 75.62% and 69.25%. Once I saw this, I decided to run one more model at 9 epoch. It gave me a training and testing accuracy of 84.36% and 70.96%.

The low performance of the model trained for 110 epochs indicates overfitting as evidenced by the massive difference in training and testing data. Furthermore, another indicator of overfitting is the massive increase in `val_loss` of 3.8016 compared to that of 1.0508 of the same models and 1.2113 of the 9-epoch models.

The superior performance of the 9-epoch model suggest that it found an improved balance between learning from the training data, testing data and the unseen data. Thus, resulting in higher testing/validation accuracy. With further fine-tuning of the hyperparameters or any modifications to the ResNet Architecture, there may be improved performance.

Reflections

In summary, the experimentation with different CNN architectures, including VGG16 and ResNet, provided valuable insights into their respective performance for the image classification tasks.

Key findings:

- *VGG16 models demonstrated good performance, with the 10-epoch model achieving the highest testing/validation accuracy of 88.71%.*
- *However, ResNet models exhibited a variation of performances. The model trained for 5 and 10 epochs did not do as well as the 9-epoch model. With the highest testing accuracy of 88.71%, it demonstrated the better generalisation compared to its counter parts.*

Problems:

- *Overfitting seemed to be prevalent in some of the VGG16 and ResNet models as the signs were there. This was in particular with the high epoch trained models as it was evident In the different in training and testing accuracies as there was a large disparity between the two.*
- *The performance of the models may have in inhibited by the limited data. As the side of the dataset may have had constraints. With an increase in data the models could potentially have an increased generalisation and learn more robust features*

Overall, the results show that both the Transfer learning models benefited from the leveraging of pre-trained weights from this ImageNet to achieve an overall better performance when image classifying. The experiment overall indicated the important of choosing the appropriate training strategies such as the varying number of epochs to achieve the optimal model performance.

Data augmentation could have been improved by implementing data augmentation techniques such as scaling, flipping, and rotation that could artificially increase the data size and further enhance the model robustness. However, in this case I wouldn't as artificially increasing the size of the image set wouldn't work well regarding marking. However, I would take it into future considerations when running more ML models.

Fine-tuning the hyperparameters to a changed learning rate, different batch size and different optimizer choices could both improve convergence and therefore improve model performance.

Early stopping different dropout rates and batch normalization could help with the issue of overfitting and mitigate any future risk. Thus, would improve the model's generalisation.

Conclusion

With this experiment concluding, it safe to say it provided valuable insight into the performance of CNN models and transfer learning via the use of VGG16 and ResNet50. With achieving competitive results, comes the issue risk of over fitting thus making it crucial to highly the use of regularisation and data augmentation techniques. Furthermore, by refining the models hyperprarameters and architecture, one could achieve a more enhansed model that could reflect onto the domains and secctors that I repviiusly spoke on in Task 1.

IV. Table of Content and Figures

Table of Contents

V. References

[1] *Esteva, A., Kuprel, B., Novoa, R.A., Ko, J., Swetter, S.M., Blau, H.M. and Thrun, S., 2017. Dermatologist-level classification of skin cancer with deep neural networks. nature, 542(7639), pp.115-118*

[2] Huang, C.L., Chen, M.C. and Wang, C.J., 2007. Credit scoring with a data mining approach based on support vector machines. *Expert systems with applications*, *33*(4), pp.847-856.

[3] Dixon, M.F., Halperin, I. and Bilokon, P., 2020. *Machine learning in finance* (Vol. 1170). New York, NY, USA: Springer International Publishing.

[4] Bojarski, M., Del Testa, D., Dworakowski, D., et al. (2016). End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*.

[5] Hastie, T., Tibshirani, R., Friedman, J.H. and Friedman, J.H., 2009. *The elements of statistical learning: data mining, inference, and prediction* (Vol. 2, pp. 1-758). New York: springer.

[6] Bishop, C.M. and Nasrabadi, N.M., 2006. *Pattern recognition and machine learning* (Vol. 4, No. 4, p. 738). New York: springer.

[7] *Obermeyer, Z. and Emanuel, E.J., 2016. Predicting the future—big data, machine learning, and clinical medicine. The New England journal of medicine, 375(13), p.1216.*

[8] Lo Piano, S., 2020. Ethical principles in machine learning and artificial intelligence: cases from the field and possible ways forward. *Humanities and Social Sciences Communications*, *7*(1), pp.1-7

[9] *Rahaman, M.A., Mahin, M., Ali, M.H. and Hasanuzzaman, M., 2019, May. BHCDR: real-time bangla handwritten characters and digits recognition using adopted convolutional neural network. In 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)(pp. 1-6). IEEE.*

[10] *Zhao, X., Sun, P., Xu, Z., Min, H. and Yu, H., 2020. Fusion of 3D LIDAR and camera data for object detection in autonomous vehicle applications. IEEE Sensors Journal, 20(9), pp.4901-4913.*

[11] *McGuinness, K., 2017. Transfer Learning [Internet]. In D2L4 Insight@ DCU Machine Learning Workshop.*