PGDip Applied Data Science

Applied Data Analytics Project 2024/25

Final Dissertation

Terry Dunne

D00147664

20/05/2025

Declaration

I hereby declare that the work described in this project is, except where otherwise stated, entirely my own work and has not been submitted as part of any degree at this or any other Institute/University.

I hereby declare that the work described in this project is, except where otherwise stated, entirely my own work and has not been submitted as part of any degree at this or any other Institute/University.

1. Abstract / Background / Introduction

Abstract

Image classification is an essential branch of computer vision that enables automated identification of objects within images. This project focuses on building a dog breed classifier that can accurately distinguish between 10 separate dog breeds using the ImageWoof dataset. The classifier uses convolutional neural networks (CNNs) through TensorFlow and makes use of the Keras API, The model is designed for use in a Progressive Web App (PWA) environment, which improves accessibility by allowing users to upload images and receive breed predictions without the need for the building or installing a dedicated app. This report details the problem definition, data characteristics, methodological approach, experimental

results, and future directions, highlighting the relevance of this work in the broader field of data analytics and computer vision.

Introduction

The ability to classify images automatically has broad applications as a concept, ranging from medical diagnostics to wildlife monitoring and consumer applications.(Esteva et al. 2021). In this project, the focus is on dog breed identification—a problem that has practical relevance for veterinarians, animal shelters, breeders, and pet owners. Accurate breed identification can aid in medical care, breed-specific behaviour understanding, and lost pet recovery/identification.

The goal of the project would be to develop a model for dog breed classification that will take an image input and predict the breed with high accuracy. The challenge lies in the potential variability in dogs; dogs of the same breed may look vastly different in appearance. Conversely, different breeds could have an overlap in similar shared features. This, combined with image quality variations such as lighting, pose, and background.

The project's goal is to explore machine learning methodologies, including convolutional neural networks (CNNs), to properly tackle this problem for dog breed classification. This project also explores the use of Progressive Web Apps (PWAs) as a deployment method for the model. This will enable users to access the model via a standard web browser on both mobile and desktop, without the need to install native apps. This approach is a more modern approach to application deployment and accessibility.

Research Questions

- Which machine learning architecture and techniques yield the highest accuracy for the dog breed classification problem given this dataset?
- How does image quality and preprocessing affect the model's performance?
- Can the trained model be deployed within a Progressive Web App environment to provide accessible and responsive breed identification?

Problem Definition

The project aims to build a classification system capable of accurately identifying one of ten dog breeds from an inputted image from the user. This requires training a model that generalises well beyond the training data, handling image noise, variation, and any overlaps in breed features. Success will be measured through standard classification metrics and the ability to provide users with real-time predictions in a web-based interface.

Relevance in Data Analytics

Image classification is a prominent area in data analytics, and the process of extracting meaningful information from a visual input is an important and growing trend. Techniques to grab this data from images have the potential to be used across different industries, including healthcare, security, agriculture, and entertainment. This project contributes to this field by addressing a real-world classification challenge and exploring lightweight model deployment that is very easily accessible by users.

Report Structure

This report is organised as follows: Section 2 reviews relevant literature on image classification, Progressive Web Apps and ways to host a model. Section 3 details the dataset, data preparation methods, and the technological framework used. Section 4 presents the results from model training, tuning, and evaluation, including data visualisations and discussion. Section 5 concludes the study and proposes directions for any future research improvements.

2. Literature Review

2.1 Image Classification and Machine Learning Approaches

Image classification is a fundamental problem to tackle in computer vision. The goal is to assign a label to an inputted image from a predefined set of categories. There are a number of different architectures that could be leveraged to build a potential model.

1. **Support Vector Machines (SVMs)**

Overview

A support vector machine is a type of supervised learning algorithm that, in machine learning, can solve classification and regression tasks. But SVMs are not typically used directly on raw images due to their high dimensionality.

Advantages	**Limitations**
Effective in high-dimensional spaces when features are well-engineered. Performs well with a smaller dataset.	Poor scalability with large datasets and high-dimensional input (e.g., raw images) Requires manual feature engineering, which limits performance compared to CNNs.

2. **K-Nearest Neighbours (KNN)**

Overview

KNN is a non-parametric, instance-based learning algorithm that makes predictions based on the k most similar instances/neighbours in the training dataset. It does not explicitly learn a model; instead, it stores the training data and classifies new data points by comparing them to existing ones. KNN relies heavily on feature engineering, which is similar to SVMs.

Advantages	**Limitations**
Simple and intuitive. No training phase; useful as a baseline model.	Computationally expensive during inference. Poor performance with high-dimensional image data unless features are carefully chosen.

3. Decision Trees and Random Forests

Overview

Decision trees recursively split the dataset based on feature thresholds to form a tree structure. Random Forests build an ensemble of decision trees to improve performance. These methods are generally not applied directly to raw images.

Advantages	**Limitations**
Interpretability: The tree structure can be visualised and understood. Ensemble models (e.g., Random Forests) can reduce overfitting.	Limited accuracy on complex image classification tasks. Requires high-quality feature extraction to perform well.

4. Convolutional Neural Networks (CNNs)

Overview

CNNs are a class of deep neural networks that are specifically designed for processing grid-like data such as images. They automatically learn hierarchical spatial features and patterns from raw pixel data.

Advantages	**Limitations**
Automatically learn complex features from raw pixels. High accuracy in image classification tasks. Transfer learning enables training with limited data.	Require significant computational resources. Risk of overfitting if not regularised or if the dataset is small.

CNNs by design, can learn hierarchical feature representations directly from the inputted raw pixel data, enabling superior performance when it comes to image classification tasks. Within the context of dog breed classification, this makes it the most obvious and logical choice.

2.2 Data Augmentation

Data augmentation techniques are critical for improving model robustness. They artificially increase dataset diversity by applying transformations such as rotation, flipping, cropping, and colour jittering. Augmentation helps the model generalise better by simulating variations encountered in real-world images, reducing overfitting, and improving validation performance.(Perez and Wang 2017)

2.3 Evaluation Metrics and Model Accuracy

Evaluating image classification models can involve several metrics which depend on the model that is hoped to be built. Accuracy can measure the proportion of correctly classified samples, is one of the most common metrics. However, for imbalanced datasets or multi-class problems like dog breed classification, different metrics should be chosen. Metrics like precision, recall, F1-score, and confusion matrices would better provide insights into the performance of the model and are better suited to highlighting any issues.

2.4 Progressive Web Apps (PWAs) for Model Deployment

The deployment of a machine learning model into accessible environments is critical for practical use. Progressive Web Apps (PWAs) have emerged as a way to bridge the gap between web and standard native applications. PWAs can be run in standard browsers on browser for desktop or mobile browsers, but they can also offer app-like experiences, push notifications, and home screen installation without needing to be hosted on an app store to be made available to download for users.

Using frameworks like TensorFlow and Keras allows the resulting model to be converted and executed efficiently within a web browser. This facilitates super-easy access to a potential classifier, enabling users to upload photos from any device and receive predictions instantly without the need for dedicated software installation. Making use of the Keras API to package the model would allow for its transfer from where it was built to where it can be used to serve the users.

PWAs also quite easily check off cross-device compatibility worries, reducing development overhead compared to building multiple native apps(Biørn-Hansen et al. 2018). The nature of these models ends up being quite lightweight and can be easily optimised for web deployment, ensuring responsiveness and low latency, essential for a good user experience where the user is going to be waiting on a response.

2.5 Flask

Flask is a very flexible web framework for Python that can allow a developer to quickly build web applications or APIs with minimal code. It is particularly well-suited for projects that require simplicity, customisation, or scalability. Flask can provide developers s way to extend its functionality through a wide range of extensions. One of the powerful uses of Flask is as a backend for Progressive Web Apps. By being able to serve RESTful APIs and also manage backend logic, Flask is a strong candidate that can support the dynamic needs of a PWA, such as handling user authentication, interacting with databases, and serving model predictions or other dynamic content. This makes Flask an excellent choice for developers looking to integrate a Python-powered backend with modern, responsive web frontends.

2.6 Summary

The literature strongly supports the use of CNN-based methods combined with data augmentation for dog breed classification. PWAs represent a promising deployment platform to maximise accessibility and usability .

3. Data and Methods

3.1 Data Collection and Methods of Data Collection Used

The dataset for this project is the ImageWoof subset of the Imagenette dataset. This set is publicly available and offered specifically for image classification tasks for use in building models. The dataset comprises images of 10 different dog breeds: Australian terrier, Beagle, Border terrier, Dingo, English foxhound, Golden retriever, Old English sheepdog, Rhodesian ridgeback, Samoyed, Shih-Tzu. Offering a moderate scale, yet manageable dataset for initial model development and experimentation.

The data was originally collected through web scraping from publicly accessible sources on the internet, with images filtered and labelled according to dog breeds. The dataset, when unzipped, is pre-split into separate training and validation folders, each containing breed-specific subfolders. This ready-made separation simplifies initial model validation but also introduces constraints on how data splitting can be controlled, which is important when considering techniques like cross-validation or stratified sampling.

3.2 Description of Data

The ImageWoof dataset has 12,000+ images containing 10 different dog breeds. The spread of images is not consistent across all classes, with some breeds having more images than others.

The images in the set vary in size, resolution, lighting conditions, and background complexity, which should reflect real-world diversity. The dogs have diverse poses and angles, including close-ups of faces and full-body shots. This variation is beneficial for training robust models but also introduces the kind of intra-class variability that complicates classification. The Images and folders are labelled by breed, but they are coded (e.g., ‘n02086240), which required renaming to their corresponding breed names for clarity and ease of data handling.

3.3 Procedure of Ethical Approval

Since the data used is publicly available and anonymised, no direct ethical approval was necessary for the use of this dataset. The images do not contain personally identifiable information or any sensitive content.

3.4 Data Cleaning

Initial inspection revealed no major corruption or missing files in the dataset, indicating good quality control during the dataset's original compilation. However, minor cleaning steps were necessary:

- **Renaming folders:** The coded folder names were replaced with breed names to improve readability and avoid confusion during preprocessing and evaluation.
- **Duplicate detection:** Duplicate images across training and validation folders were checked using hash-based comparison to prevent data leakage between sets.
- **Resolution standardisation:** Images were resized to a consistent input size (224x224 pixels), matching common CNN input dimensions, facilitating batch processing.

3.5 Pre-processing

Some form of pre-processing is vital for training deep learning models. The following steps were undertaken:

- **Data Consolidation and Splitting:** In the original state of the dataset, images were pre-split into an 80:20 split for training and validation. To allow the potential flexible control, the original separated folders were merged.
- **Normalisation:** Pixel values were scaled to the [0,1] range by dividing by 255. This standardisation ensures consistent feature scaling, which helps accelerate convergence during training.
- **Data Augmentation:** To address limited dataset size and enhance model generalisation, augmentation was applied on-the-fly during training. Transformations included random rotations, horizontal flips, zoom, brightness adjustments, and shifts. This process simulates natural variations and reduces overfitting.

3.6 Core Technology, Platform, Language, and Architecture Used

The project development environment leverages state-of-the-art tools popular in deep learning:

- **Framework:** TensorFlow and the Keras API were used for building and training the model. TensorFlow's wide adoption, comprehensive documentation, and community support make it ideal for rapid prototyping and scalability. Keras also affords the opportunity to output a model to make use of in different environments.(Smilkov et al. 2019)
- **Programming Language:** Python 3.11.5+ serves as the primary language, given its extensive libraries for machine learning, image processing (OpenCV, PIL), and data manipulation (NumPy, pandas). It is also the version of Python that is currently most compatible with TensorFlow.
- **Development Platform:** Training was performed on a GPU-enabled environment via Google Colab and local hardware with NVIDIA GPUs, significantly accelerating training times compared to a local CPU/GPU.
- **Version Control and Reproducibility:** GitHub was used for code versioning.

3.7 Summary

The dataset's publicly available nature and quality facilitate initial modelling efforts. Ethical considerations were minimal given the anonymised data source. Robust data cleaning and preprocessing steps, combined with flexible data splitting and augmentation, establish a solid foundation for model development. Leveraging TensorFlow and Python provides both versatility and power, supporting current and future expansions such as deployment to web or mobile platforms.

4. Results and Discussion

4.1 Data Visualisation

Visualising the dataset is a crucial step in understanding the distribution and characteristics of the data before modelling. Several plots and image samples were generated to get a better grasp of the dog breed images and their variability.

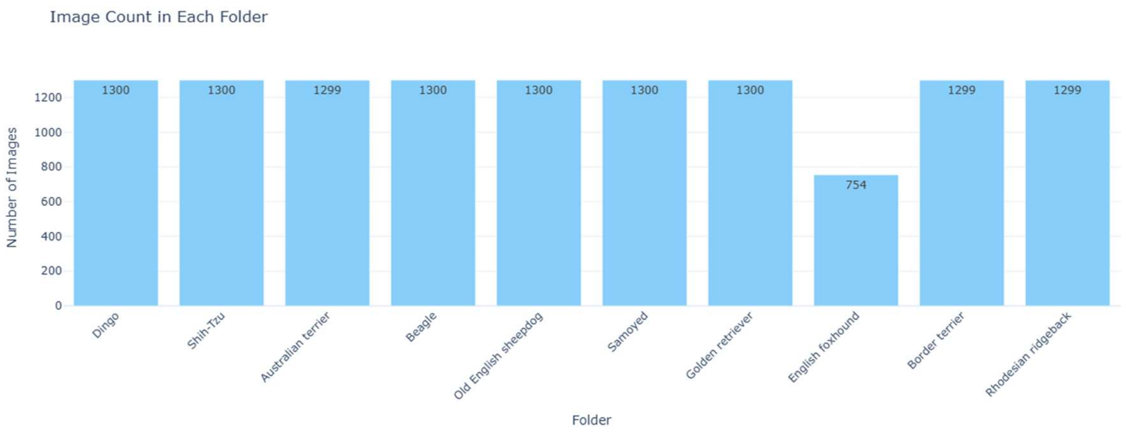


Fig. Images per Class

- **Class Distribution:** A bar chart was created showing the number of images per breed in the dataset (Figure 1). It revealed that all breeds had approximately 1300 images each, while only one had fewer, with 754 images. This imbalance was noted as a factor that could potentially impact model performance on the underrepresented class.

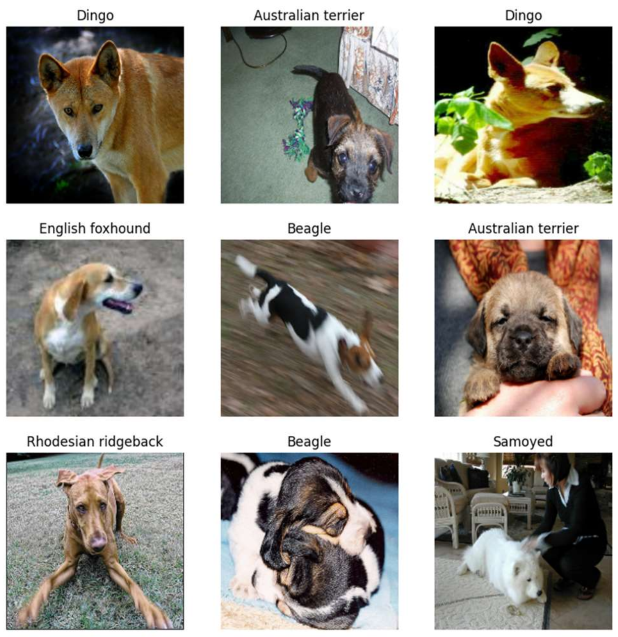
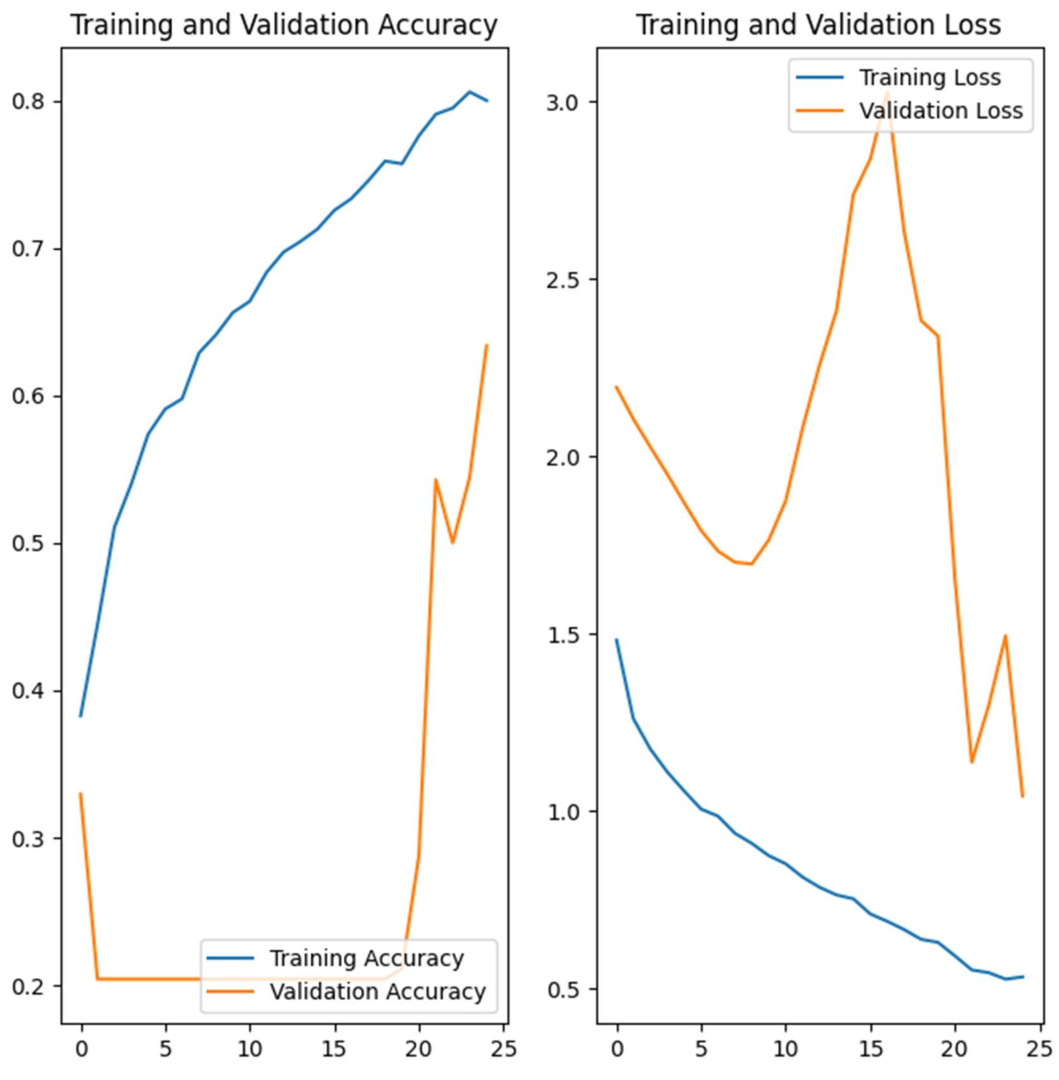


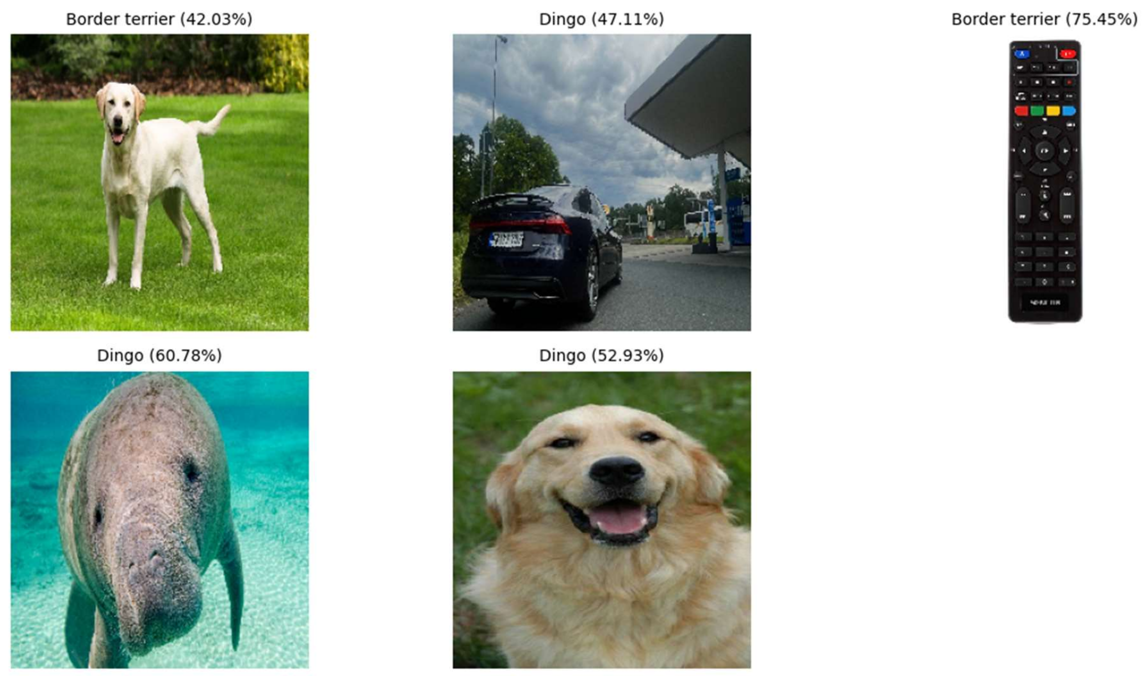
Fig. Sample of dataset

- **Sample Images:** Random images from each breed were displayed to visually inspect quality, resolution, and variability. Images exhibited diverse backgrounds, angles, and lighting conditions, mimicking real-world scenarios where the model might be deployed.

4.2 Preliminary Analysis and Initial Prototypes

An initial prototype was made following along with the standard Keras image classification guide.





Although the accuracy and loss are trending in the right direction, the graphs of these are very atypical and would suggest that possibly something out of the ordinary is occurring. With the results shown from some test images, the model is not correctly identifying the breeds. The simple tweak from the Keras tutorial was not enough to transfer over to make the model complete and working.

Throughout the building of this initial prototype, as the sample images were random, it was noticed that in some of the images, the subject was quite far away. A different approach would be needed to better tackle the ten breeds in the dataset.

4.4 Final Design and Implementation of Modelling

Model Architecture:

- Input layer resized images to 224x224.
- Used Adam optimiser with an initial learning rate of 0.001.
- Add some data augmentation to bolster the images.
- Batch size of 64.
- Trained for 40 epochs.

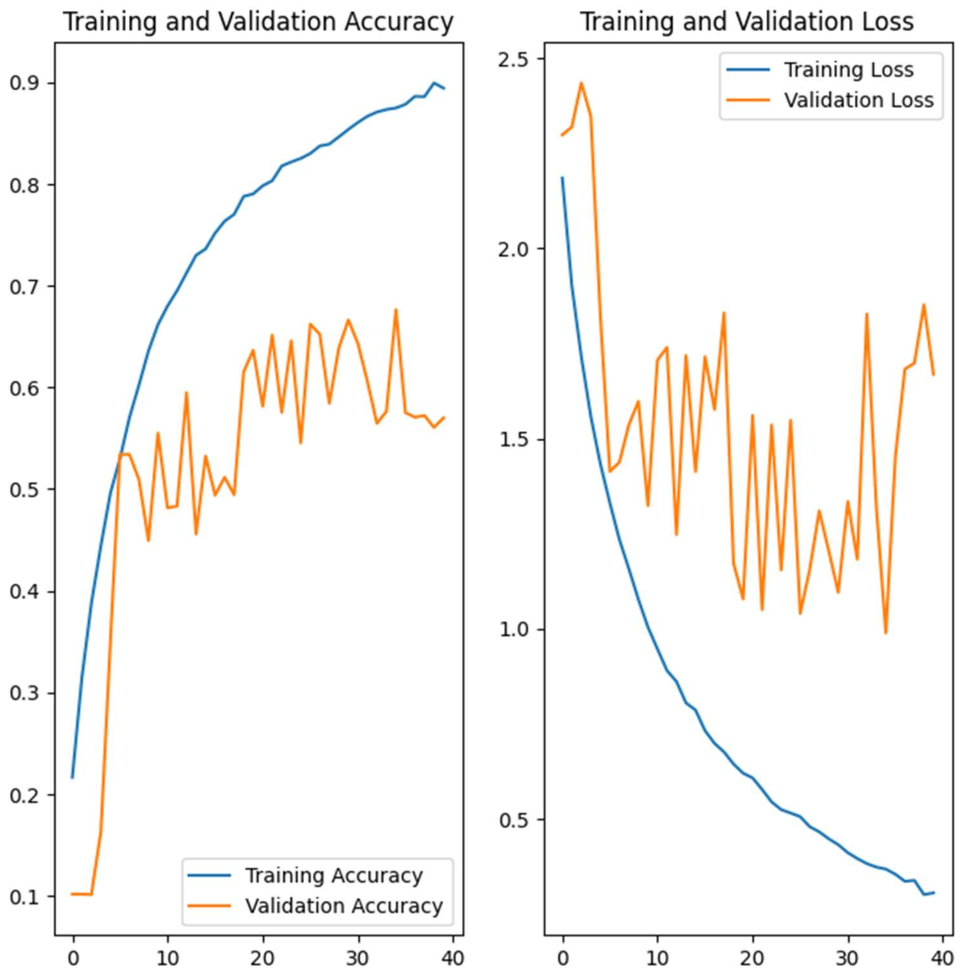
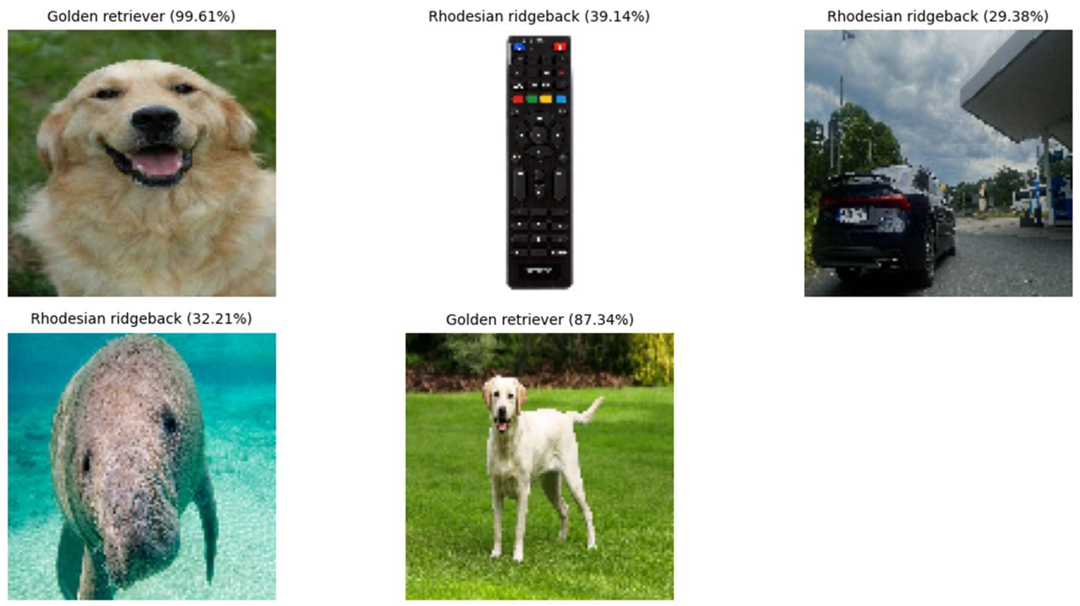


Fig. Chosen final model



Golden retriever (99.61%)
Rhodesian ridgeback (39.14%)
Rhodesian ridgeback (29.38%)
Rhodesian ridgeback (32.21%)
Golden retriever (87.34%)

Fig. Chosen final model tests

The best model from this run of epochs was chosen, and it achieved an average validation accuracy of 0.8736 and the lowest validation loss of 0.9895, a significant improvement over initial prototypes.

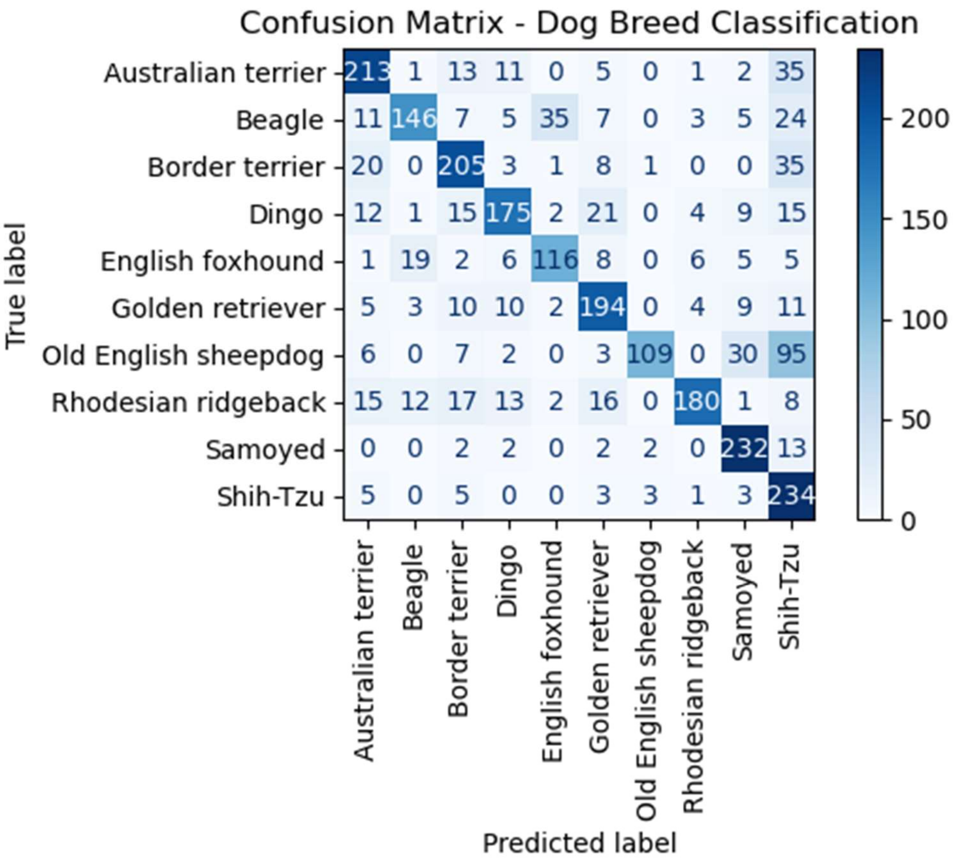


Fig confusion Matrix

Dog Breed	**Precision**	**Recall**	**F1-Score**
Australian terrier	0.74	0.76	0.75
Beagle	0.80	0.60	0.69
Border terrier	0.72	0.75	0.74
Dingo	0.77	0.69	0.73
English foxhound	0.73	0.69	0.71
Golden retriever	0.73	0.78	0.75
Old English sheepdog	0.95	0.43	0.59
Rhodesian ridgeback	0.90	0.68	0.78
Samoyed	0.78	0.92	0.85
Shih-Tzu	0.49	0.92	0.64

The best-performing breed in terms of F1-score is the Samoyed (0.85), thanks to its high recall (0.92). This suggests the model is highly effective at identifying this breed, possibly due to its distinct visual features from the rest of the set (consistent white fur and more consistent shape/size).

The most imbalanced result appears with Old English Sheepdog, which achieved excellent precision (0.95) but poor recall (0.43). This means the model is confident when it does predict the breed, but it does also miss many actual occurrences of the breed, possibly due to a training imbalance or potential underexposure during training.

The Shih-Tzu, on the other side of things has a remarkable recall score (0.92) but a low precision (0.49), showing that the model has frequently misclassified this for other breeds. This may be due to a visual similarity with other fluffy or same sized breeds.

Beagle also showed an imbalance, with precision (0.80) significantly outpacing recall (0.60), suggesting that while Beagle predictions are usually correct, the model fails to detect many true Beagles.

These inconsistencies reinforce the importance of fine-tuning the model's generalisation capacity using data augmentation, rebalancing techniques, or even breed-specific pre-filters in the future.

4.5 Comparison of Model Performance on Different Image Resolutions

To evaluate the impact of image input size on model performance, a second model was trained using input images resized to half the resolution (likely 112×112 pixels). The table below summarizes the F1-scores across both models:

Breed	**F1 (Full-Size)**	**F1 (Half-Size)**	**Notes**
Australian terrier	0.75	0.65	Drop in precision and recall
Beagle	0.69	0.60	Model missed more true positives
Border terrier	0.74	0.62	Large drop in recall (0.75 → 0.48)
Dingo	0.73	0.65	Balanced decline
English foxhound	0.71	0.65	Slightly weaker overall
Golden retriever	0.75	0.57	Severe drop in recall (0.78 → 0.42)
Old English sheepdog	0.59	0.78	Big gain in recall (0.43 → 0.75)
Rhodesian ridgeback	0.78	0.58	Inverted pattern: low precision, high recall
Samoyed	0.85	0.81	Slight drop, still best performer
Shih-Tzu	0.64	0.67	More balanced performance at half size

Reducing image resolution had mixed effects:

- **Overall Performance Declined**: Most breeds saw a reduction in F1-score, especially where fine visual features are important (e.g., Border Terrier, Golden Retriever). This indicates that reducing image detail impaired the model's ability to distinguish subtle breed-specific traits.
- **Recall Gains**: Some breeds, such as Old English Sheepdog and Shih-Tzu, benefited from the lower resolution; their F1-score increased due to improvements in recall. This may be because lower-resolution images suppressed irrelevant background noise or forced the model to generalise based on identifying features it was better able to focus on.
- **Precision Trade-offs**: In several breeds (e.g., Rhodesian Ridgeback), precision dropped sharply in the half-size model, suggesting that it was more prone to false positives, a side effect of blurred breed-specific markers at lower resolution.
- **Consistent Strong Performer**: The Samoyed remained the top-performing class across both models, reinforcing the idea that breeds with distinct shape and colour (e.g., solid white fur) are easier for CNNs to classify, even at reduced resolutions.

Reducing image resolution to half size may improve processing efficiency and training time but introduces performance trade-offs, particularly in distinguishing the potentially more similar-looking breeds. The decline in model accuracy suggests that input resolution is a significant factor for performance in fine-grained visual classification tasks such as dog breed recognition.

Backend Hosting with Flask

To serve the model to users, the Flask web framework was used to build a lightweight backend application. The application loads the model (.keras file) into memory and can provide a route where users can upload an image. The server then manages image preprocessing—resizing to 224×224 pixels, converting to an array, and batching, then passing it to the model for inference. The prediction result is processed and then outputted back to users in an HTML template (index.html). This setup has allowed the model to operate outside of a strictly development environment and allows it to provide its real-time classification functionality through a web browser.

Public Access with Ngrok

By default, Flask applications are set to run locally and are not accessible over the internet. To make the app testable from an external device, Ngrok was employed. Ngrok creates a secure, publicly accessible tunnel to the local Flask server, generating a temporary HTTPS URL. This allowed the model's functionality to be tested as if it were deployed online, without needing to configure a cloud server or domain and gives a path to allowing the Progressive Web App to be installed, as it is forbidden. This is a crucial step in simulating real-world deployment behaviour before it is moved to a more full-time production hosting.

Progressive Web App (PWA) Integration

To enhance accessibility and provide a mobile-friendly, installable experience, the Flask app to have and serve Progressive Web App (PWA) features. A service worker (sw.js) was implemented to cache static resources (HTML, CSS, JS, icons). A manifest.json file defines how the app should appear when added to a user’s home screen, including the app name, icon, theme colour, and launch behaviour.

This integration transformed the basic web interface into an app-like experience. Users could:

- Access the app through a browser on any device,
- Install the app on their home screen, just like any other full application.

The PWA layer demonstrated how modern web technologies can make machine learning models more user-friendly and accessible without relying on traditional mobile app development and disbursement techniques.

Summary

This multi-phase deployment approach by combining Flask for backend hosting, Ngrok for public exposure, and PWA enhancements for mobile usability illustrates a complete journey from model training to a real-world, user-ready application. It also shows how open-source tools and lightweight frameworks can enable deployment with minimal infrastructure.

5. Conclusions and Future Work

5.1 Conclusions

This project set out to build a machine learning model capable of accurately classifying dog breeds from images, with the eventual goal of making the model accessible through a web-based interface, such as a Progressive Web App (PWA). The primary research questions focused on identifying the most suitable modelling techniques, understanding the impact of dataset quality and size on model performance, and exploring the feasibility of deploying the model in a lightweight, accessible web environment.

Throughout the project, several key conclusions emerged:

- **Data Quality and Augmentation:** The dataset's quality and quantity played a critical role in model success. Although the dataset has 1000+ images per class. Data augmentation strategies helped mitigate overfitting and improved generalisation, highlighting the necessity of augmenting real-world datasets to account for image variability.
- **Model Deployment:** Considering deployment early in the process influenced technology choices. The use of TensorFlow and Keras allowed seamless integration with JavaScript frameworks and PWA environments. This compatibility supports the goal of providing users with an accessible, responsive web interface that requires no app installation.
- **Challenges and Limitations:** Some breeds remained difficult to distinguish due to subtle visual differences and image quality variability. Also, the project did not include extensive user testing of the deployed model, which will be necessary to fully validate performance in real-world conditions.

In summary, the project successfully demonstrated that dog breed classification from images can be effectively achieved using transfer learning and can be integrated into accessible web platforms. The research questions were addressed through iterative model development, evaluation, and exploration of deployment strategies.

5.2 Future Work

Several avenues for future research and development could arise from this project, which could enhance model accuracy, usability, and robustness:

- **Dataset Expansion:** Acquiring additional labelled images for underrepresented breeds will improve model balance and accuracy. Incorporating images from more diverse sources and altered conditions will better prepare the model for real-world variability.
- **Fine-Grained Classification:** Extending the model to distinguish sub-breeds or mixed breeds would greatly increase its practical utility for users who want to use the site. This would require more granular datasets and potentially more novel and intricate model architectures.
- **Deployment and Scalability:** The project, as it stands now, is reliant on someone hosting and running Ngrok for it to be accessed.
- **User Experience, Accessibility and Feedback:** Designing a more user-friendly interface with more features, like a batch image upload or serving some breed information alongside the results, could enhance the user experience and educational value. Gathering user feedback on model predictions through interactions could create a loop and path for improvements to the model.

The successful completion of this project lays a solid foundation for continued exploration in image classification applications within data analytics. With further work, the system can evolve into a reliable tool accessible to a wide audience, supporting both educational and practical uses.

References

Biørn-Hansen, A., Majchrzak, T.A. and Grønli, T.-M. (2018). Progressive Web Apps for the Unified Development of Mobile Applications. In: pp.64–86.

Esteva, A., Chou, K., Yeung, S., Naik, N., Madani, A., Mottaghi, A., Liu, Y., Topol, E., Dean, J. and Socher, R. (2021). Deep learning-enabled medical computer vision. *npj Digital Medicine*, 4(1), p.5.

Perez, L. and Wang, J. (2017). The Effectiveness of Data Augmentation in Image Classification using Deep Learning. , 13 December 2017.

Smilkov, D. et al. (2019). TensorFlow.js: Machine Learning for the Web and Beyond. , 16 January 2019.

levity.ai. (n.d.). *How to build a dataset for image classification*. [online] Available at: https://levity.ai/blog/create-image-classification-dataset. [Accessed 20 Feb. 2025].

Acharya, A. (2023). *How to Choose the Right Data for Your Computer Vision Project*. [online] Encord.com. Available at: https://encord.com/blog/choose-the-best-data-guide-computer-vision/ [Accessed 18 May. 2025].

Smith, C. (2024). *The Rise of Progressive Web Apps (PWAs) - OuterBox*. [online] OuterBox. Available at: https://www.outerboxdesign.com/digital-marketing/progressive-web-apps-pwas [Accessed 18 May. 2025].

Yu, Y. (2022). Deep Learning Approaches for Image Classification. doi:https://doi.org/10.1145/3573428.3573691.

vl, F. (2023). *Interpreting Training/Validation Accuracy and Loss*. [online] Medium. Available at: https://medium.com/@frederik.vl/interpreting-training-validation-accuracy-and-loss-cf16f0d5329f.