**PGDip Applied Data Analytics Project Interim Report**

Terry Dunne

D00147664

29/05/2025

# 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. Using Keras, 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 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. 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 generalizes 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 accessed by users.

**Research Methodologies Used**  
The project uses supervised learning with convolutional neural networks, transfer learning from pretrained models, and data augmentation for enhanced generalization. Experimental methodology includes dataset preprocessing, training-validation-testing split, hyperparameter tuning, and performance evaluation through accuracy, loss, and confusion matrices.

**Report Structure**  
This report is organised as follows: Section 2 reviews relevant literature on image classification, transfer learning, and Progressive Web Apps. 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 visualizations and discussion. Section 5 concludes the study and proposes directions for future research.

# 2. Literature Review

**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 several different architectures that could be utilised to build a potential model.

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

**Overview**

A support vector machine (SVM) is a type of supervised learning algorithm that is used in machine learning to solve classification and regression tasks. 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.. |

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

**Overview**

KNN is a non-parametric, instance-based learning algorithm that classifies an image by a majority vote of its nearest neighbours in its given feature space. Like SVMs, KNN relies heavily on feature engineering.

|  |  |
| --- | --- |
| **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. |

1. **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. |

1. **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 through convolutional layers.

|  |  |
| --- | --- |
| **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 are able to 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, CNNs are well-suited due to their ability to extract discriminative features despite intra-class variability and inter-class similarity.

**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.

**2.4 Evaluation Metrics and Model Accuracy**

Evaluating image classification models involves several metrics. Accuracy, which is the proportion of correctly classified samples, is the most common metric. However, for imbalanced datasets or multi-class problems like dog breed classification, additional metrics, which could be precision, recall, F1-score, and confusion matrices, would provide deeper insights into the performance of the model.

Cross-validation and the use of a separate validation set are essential to detect overfitting. The loss function, commonly categorical cross-entropy for multi-class problems, quantifies the error between predicted probabilities and true labels. Monitoring loss and accuracy during training enables fine-tuning of hyperparameters like learning rate, batch size, and network depth.

**2.5 Progressive Web Apps (PWAs) for Model Deployment**

The deployment of machine learning models in accessible environments is critical for practical use. Progressive Web Apps (PWAs) have emerged as an innovative solution bridging the gap between web and native applications. PWAs run in standard browsers but offer app-like experiences, including offline access, push notifications, and home screen installation without app store downloads.

Using frameworks like TensorFlow.js allows models trained in Python/TensorFlow to be converted and executed efficiently within a web browser. This facilitates easy access to dog breed classifiers, 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.

PWAs also address platform fragmentation by providing cross-device compatibility, reducing development overhead compared to building multiple native apps. The lightweight nature of models optimised for web deployment ensures responsiveness and low latency, essential for a good user experience.

Flask is a lightweight and flexible web framework for Python that allows developers to quickly build web applications or APIs with minimal boilerplate code. It is particularly well-suited for projects that require simplicity, customisation, and scalability. Flask provides and allows developers 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.

**2.7 Summary**

The literature strongly supports the use of CNN-based transfer learning methods combined with data augmentation for dog breed classification. PWAs represent a promising deployment platform to maximise accessibility and usability. However, careful attention must be paid to dataset quality, model optimisation, and user experience to build an effective solution.

# 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.

An additional dog breed was added to address one of the research questions. Rather than seeking out another dataset, A script was made to pull images from Reddit in a specific dog breed subreddit.

**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.

From the script that was run, it resulted in 1,000 images of an additional dog breed that were added to the model after the initial build of the model

**3.3 Procedure of Ethical Approval**

Since the data used is publicly available and anonymized, 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 standardization:** 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 separate 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.
* **Label Encoding:** Breed labels were encoded into a numerical format suitable for multi-class classification. One-hot encoding was used to convert the 10 breed classes into vectors, compatible with categorical cross-entropy loss functions.
* **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.
* **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 compatible with TensorFlow
* **Development Platform:** Training was performed on a GPU-enabled environment via Google Collab and local hardware with NVIDIA GPUs, significantly accelerating training times compared to a local CPU/GPU.
* **Model Architecture:** Custom CNN models were initially developed from scratch, drawing inspiration from classical architectures such as VGG and ResNet. Later phases will explore transfer learning by fine-tuning pretrained models like ResNet50 or MobileNetV2, balancing accuracy with computational efficiency.
* **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.

|  |
| --- |
|  |
|  |

* **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.

|  |
| --- |
|  |
|  |

* **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.

A graph of training and validation loss

AI-generated content may be incorrect.

A collage of a dog and a car

AI-generated content may be incorrect.

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 with early stopping based on validation loss.

**4.5 Parameter Tuning, Evaluation, and Testing**

* **Accuracy:** The best model achieved an average validation accuracy of 0.8736 and a lowest validation loss of 0.9895, a significant improvement over initial prototypes.

A screenshot of a graph

AI-generated content may be incorrect.

* **Confusion Matrix:** Analysis revealed that certain breeds with subtle visual differences (e.g., Border Collie vs. Australian Shepherd) still caused some misclassifications, indicating potential areas for dataset enrichment or model refinement.

| **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 macro average (unweighted mean of scores across all classes) and the weighted average (accounts for support/class frequency) both converge at approximately 0.76 precision, 0.72 recall, and 0.72 F1-score. This indicates generally balanced model performance, though slightly favouring precision over recall.

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

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 misses many actual occurrences, possibly due to a training imbalance or potential underexposure during training.

Conversely, the Shih-Tzu had a very high recall score (0.92) but a low precision (0.49), indicating that the model has frequently misclassified other breeds as a Shih-Tzu. This may be due to a visual similarity with other fluffy or light-coloured 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-Dominated 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 generalize based on identifying features.
* **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, likely a side effect of blurred breed-specific markers at lower resolution.
* **Consistent Strong Performer**: **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.

# 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 contained thousands of 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 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 diverse sources and conditions will better prepare the model for real-world variability.
* **Advanced Architectures:** Experimenting with state-of-the-art architectures such as EfficientNet, DenseNet, or vision transformers could yield higher accuracy and better feature extraction capabilities. Ensemble learning combining predictions from multiple models may also boost performance.
* **Fine-Grained Classification:** Extending the model to distinguish sub-breeds or mixed breeds could increase its practical utility. This would require more granular datasets and potentially novel model architectures.
* **Explainability and User Feedback:** Implementing explainable AI techniques could help users understand model decisions, increasing trust and engagement. Gathering user feedback on model predictions through the web interface could create a feedback loop for continuous model improvement.
* **Deployment and Scalability:** Exploring cloud deployment options such as TensorFlow Serving or AWS SageMaker would support scalable, low-latency inference. Additionally, optimizing the model for edge deployment on mobile devices could reduce dependence on network connectivity.
* **User Experience and Accessibility:** Designing a more user-friendly interface with features like batch image upload, confidence scores, and breed information could enhance the user experience and educational value.

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**

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 20 Feb. 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 20 Feb. 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.

‌ Team, K. (n.d.). *Keras documentation: Image classification from scratch*. [online] keras.io. Available at: https://keras.io/examples/vision/image\_classification\_from\_scratch/.

‌

https://www.geeksforgeeks.org/convolutional-neural-network-cnn-architectures/

https://medium.com/@imjeremyhi/notable-cnn-architectures-and-how-they-work-dd46dea65671