Artificial Intelligence Assignment Challenge “Facial Recognition - Say What You See” - Shaheer Hussain

Table of contents :

Planning for an AI agent and machine learning model

In today's age of artificial intelligence (AI), the demand for image recognition models has become essential across industries from healthcare to retail. My document will outline the steps involved in creating an AI model for image recognition. It encompasses activities such as data preparation, categorization, training, validation, model testing, AI agent design, development and assessment. Each step contributes to ensuring the effectiveness and precision of the AI model in achieving classification results.

Data Preparation :

The initial stage of setting up an AI model for image recognition focuses on data preparation. This includes organising the dataset by standardising images and normalising data to ensure consistency and accuracy during training. The dataset may consist of images categorised into groups like cats, dogs or other objects. Proper organisation of the dataset is crucial for training and validation processes. For my AI agent, I collected data from already made datasets from a reputable website called Kaggle (https://www.kaggle.com/), In particular, I made use of an extremely detailed cat image dataset uploaded by Chris Crawford (https://www.kaggle.com/datasets/crawford/cat-dataset) which has **over 8000 images** and the similarly detailed stanford dogs dataset (https://www.kaggle.com/datasets/jessicali9530/stanford-dogs-dataset?resource=download) which has **20,580 images.** Finally, I also chose to make use of a human faces dataset (https://www.kaggle.com/datasets/ashwingupta3012/human-faces) which has **over 7200 images**. I chose to use these datasets as they are quite simple to evaluate in terms of their data quality and integrity but also it would make the process of data preparation much easier as these datasets are commonly used for projects like image classification.

Categorization :

Following organisation is the categorization phase, This involves assigning labels to each image based on its category. Creating subsets within the dataset for training, validation and testing purposes is essential. Accurate labelling is vital for teaching the AI model to recognize categories dividing subsets and aids in evaluating the model's performance at various development stages. For my AI models which I trained, I chose to use very simple labels that generally indicated whether a subject was in or not in the image, for example, two labels could be “Dogs” to indicate there are dogs in the image or “Not dogs” to indicate that there are no dogs in the image, of course with the cats and humans models they would follow the same structure.

Training :

Training an AI model entails considering parameters and hyperparameters.
Factors such, as batch size, learning rate, number of epochs and techniques to prevent overfitting and underfitting are considerations. The batch size determines how many samples are processed before updating the model parameters while the learning rate controls the size of steps taken during parameter updates. It is important to choose values for these hyperparameters to improve the model's performance and shorten the training time. I am using “Google’s Teachable Machine” to train AI models which my code will then access, I have chosen to keep the batch size as “16” as the tool mentions that this option does not need to be tweaked in order to affect the performance of the model, However, I have increased the Epochs to “100”, this is because the AI model will be more trained on the sample images given which can result in the model being more accurate, I also have kept the learning rate as “0.001” which is the default, I did not feel confident about increasing the number as this may affect the quality of the AI models results when trying to predict or classify images.

Validation :

Validation is a part of the training process as it evaluates how well the model performs on a subset of data that was not used in training. This evaluation helps gauge how effectively the model generalises and identifies any issues related to overfitting or underfitting. Various validation methods like cross-validation can be employed to ensure the model's reliability.

For validation I will be taking images from google images as they will be images not included in the data set, I will also test against images of my own however I will not include this in my Figures for privacy reasons.

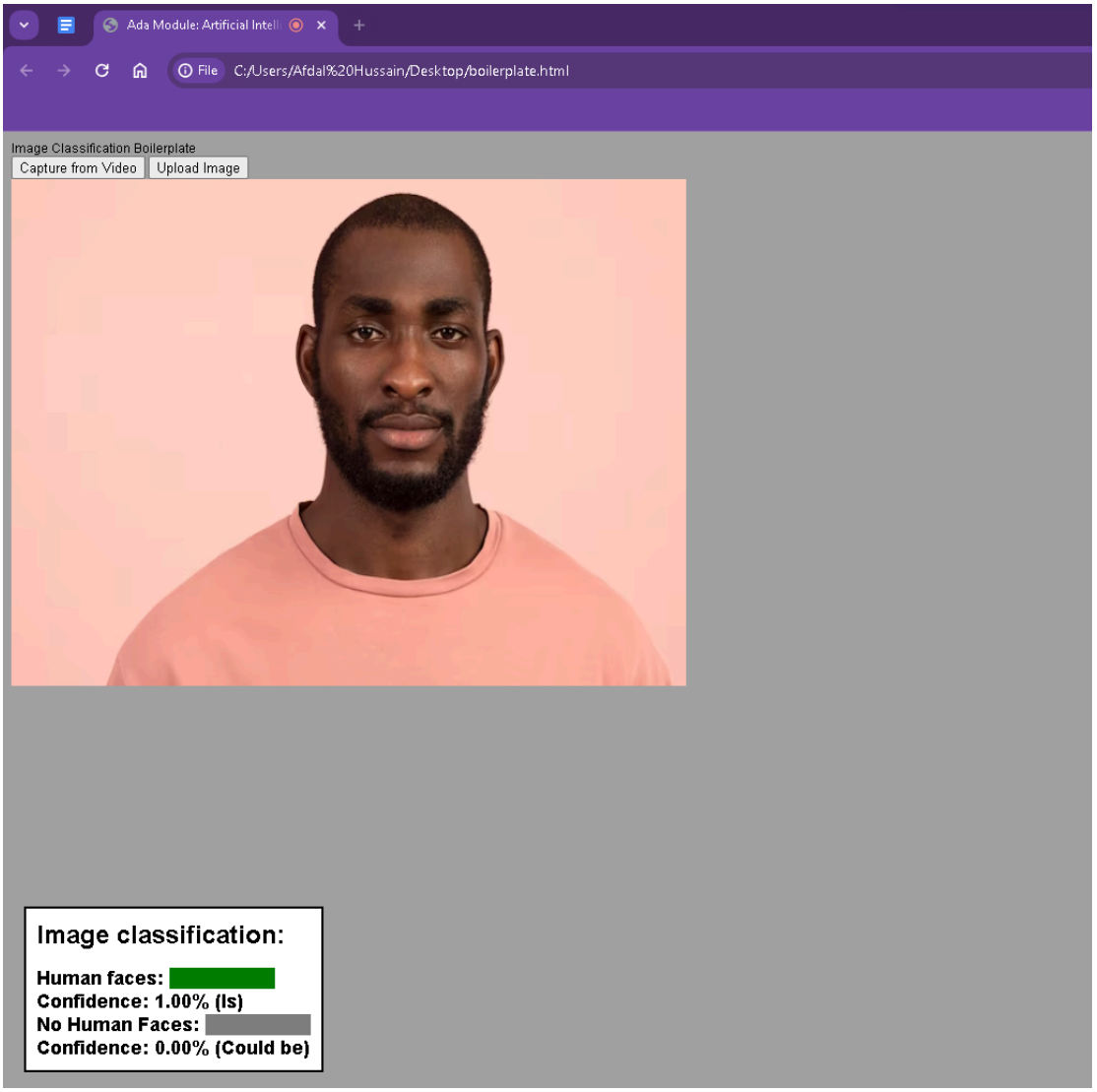


Figure 1 : Validation against google image of human portrait (Human model)

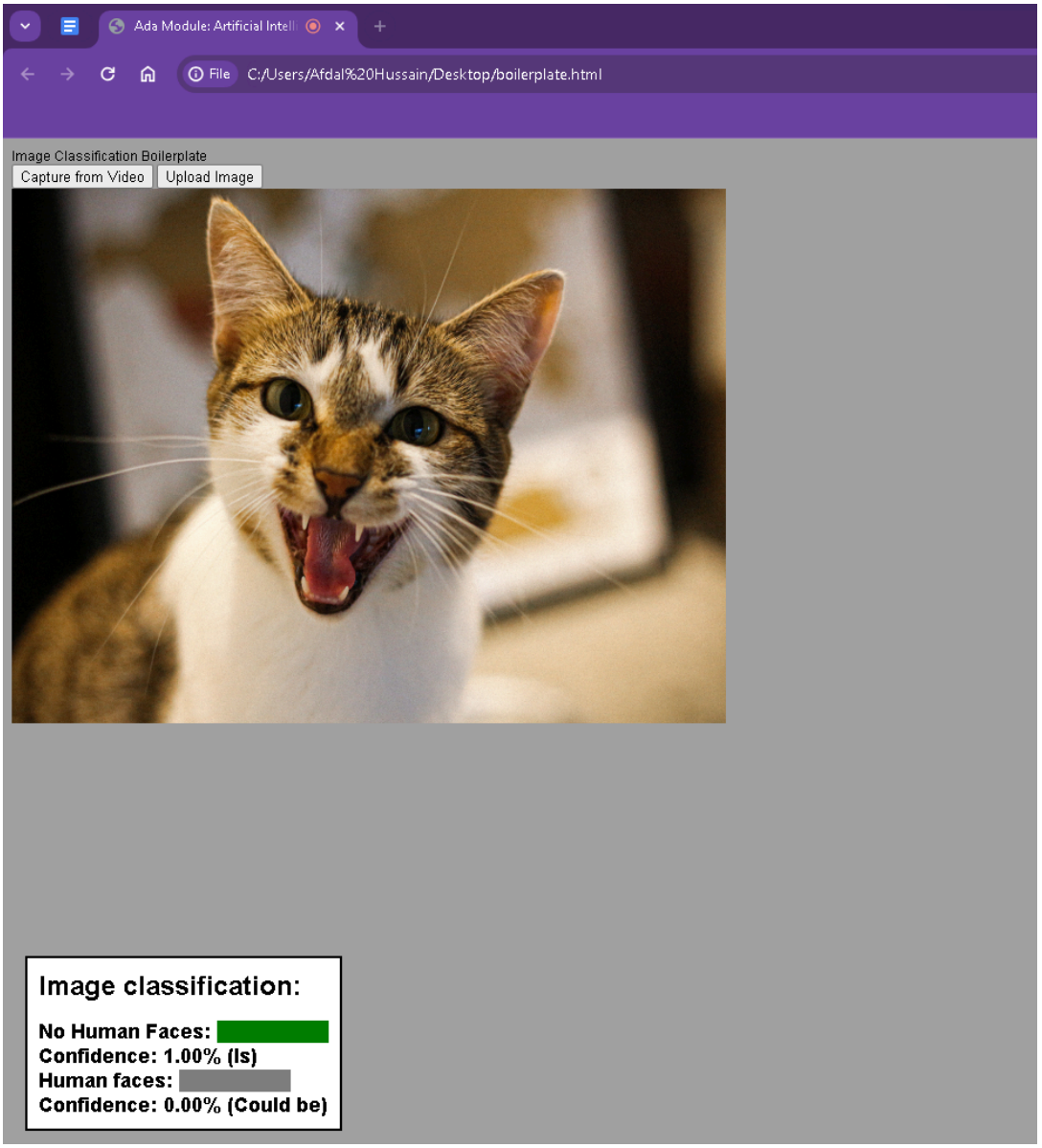


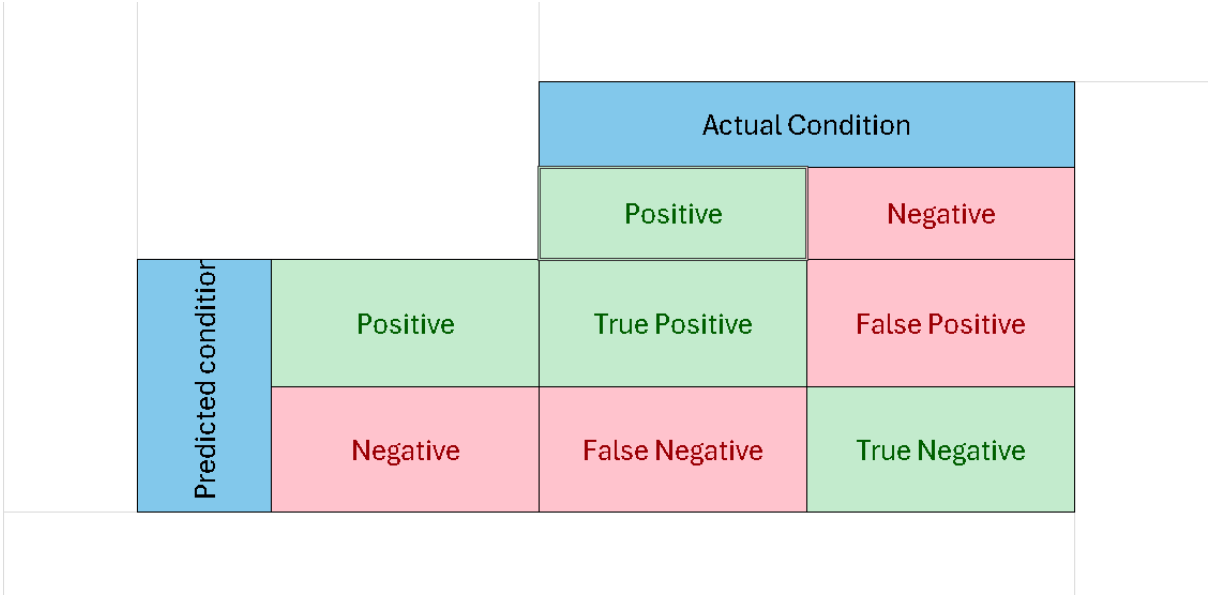
Figure 2: Validation against Google image of cat portrait (Human model)

Figures 1 and 2 make use of the Human AI model I have created which aims to classify images to detect whether there is a human present in the image or not, as mentioned earlier I have taken images from Google to evaluate how well the model performs on a subset of data that was not used in training. In Figure 1 we can see that my AI model believes that there is a human present which is correct and in Figure 2 we can conclude that the AI model believes that there are no humans present which is also correct as the image uploaded is of a cat looking into the camera.

Model Testing. Performance :

After training and validation, the next step involves testing the model to evaluate its performance using metrics such as True Positives (TP) True Negatives (TN) False Positives (FP) and False Negatives (FN). These metrics provide insights into how the model performs in real-world scenarios by considering its accuracy, precision, recall and F1 score. Additionally, tools like confusion matrices and ROC curves can be valuable, for visualising how effectively the model is functioning.

For my confusion matrix I will be using the following format :



Model Design (15% -suggest: 500 words)

In this review, I will delve into an assessment of my AI model and agent focusing on their intended use, classifiers, design and data. The intricate relationship between these elements is crucial for the success of any machine learning project requiring analysis to ensure performance and dependability.

First of all, I have created and trained 3 models using “Google's teachable machine”, the fist model is trained on over 7200 images of human faces to classify whether an image has a human present or not, and the second model is trained on over 20000 Dog’s faces to classify whether an image had a dog present or not and finally the third model is trained on over 9000 images of cat’s faces to classify whether an image has cats present or not.

An AI agent is a computer program which makes decisions using its environment and ultimately takes actions to meet certain criteria or goals.**(Bansall, 2019)** The purpose of my AI agent is to capture a webcam feed or an image file that a user uploads and use the trained models to then classify the webcam feed or uploaded image file based on the classes created in the models. To make it simple, if I use my AI model on human faces as an example, the classes within this model are “Human faces” and “No Human faces”, this means the AI agent will classify the user webcam or uploaded image using the trained human faces model, making use of the two classes to distinguish whether the input has a human face present or not.

To build and train my models I made use of a web application called “Teachable machine” which is made by Google, this is a tool which is designed to make machine learning more accessible by allowing users to easily train their machine learning models without any extensive coding knowledge and skills, Google does this via a very intuitive and easy to understand interface which goes onto my

next couple of points, I chose teachable machine because of the following points:

- User-friendly UI - The teachable machine has an incredibly easy-to-understand interface which makes the task of creating machine learning models much easier for everyone.
- Versatility in function - Teachable machine don't just do image classification which is what I am doing for this assignment but they also allow you to sound classification and pose estimation, potentially giving me more options for this assignment.
- Accessibility - As a result of Teachable machine being a web-based application, it is more accessible for me as I can use different devices to train my model which can make the process even more efficient.
- Classification - With the Image classification, Teachable machine has an in-built live classifier which allows you to manually test just how accurate your model is going to be, this is important as accuracy determines the effectiveness of a model.

However, there are some cons to Google's teachable machine, for example :

- Complexity - Teachable Machine is great, for beginners and simple projects. It may not be the choice for more complex machine-learning tasks that need advanced techniques or extensive datasets. Additionally compared to machine learning platforms Teachable Machine has some limitations in terms of customization. Users might feel constrained when trying to apply algorithms or techniques due to these limitations.
- Dependency - Another point to consider is the tool's dependency on Google since it is developed by the tech giant. This could raise concerns about privacy and data usage for users who are cautious about sharing their data with large companies like Google.
- Limited export formats - Although Teachable Machine allows models to be exported in formats it may not cover all options that developers require for their projects.
- Performance - the performance of models trained using Teachable Machine might not match those trained with tools and techniques. This can be particularly noticeable, in tasks that demand accuracy or precision.

Dog model (Dogs present - No Dogs present) :

Data Points : Over 20,000

Reasoning :

Dogs present a range of breeds, sizes, colours and orientations leading to diversity in their facial characteristics. Utilising a varied dataset ensures that the model acquires capabilities to identify dog faces amidst different variations. Incorporating images enhances the model's ability to generalise effectively to data. By exposing the model to a plethora of examples it can grasp dog features and patterns thereby reducing the risk of becoming too specialised in particular breeds or angles. Dog faces exhibit expressions, angles and lighting conditions. A diverse dataset enables the model to

learn how to discern dogs under circumstances and poses thereby improving its capacity to adapt to real-world scenarios.

Number of Categories : 2 (Presence of Dogs, Absence of Dogs)

Reasoning :

This model focuses on classifying images based on whether they feature a dog's face. The two distinct categories, "Presence of Dogs" and "Absence of Dogs " allow for sorting into these groups. By framing the task as a classification challenge the training process becomes more straightforward and aids in understanding how the model operates. Its primary goal is to differentiate between images with dogs and those without facilitating performance assessment and prediction interpretation.

"Dogs, in the Picture" Category :

Number of Data Points : 20,000 (for the Dog Face Model)

Reasoning :

This category showcases images featuring dogs' faces. The inclusion of 20,000 images ensures that the model learns a range of features related to dog faces, such as breed variety, colours, sizes and angles. Dogs display expressions, poses and lighting conditions. Training on a dataset allows the model to grasp this diversity effectively improving its accuracy in recognizing dog faces in settings. By including several images in this category we reduce the risk of overfitting and enhance the model's ability to generalise well when encountering dog-face scenarios. This boosts its performance in real-life situations.

"No Dogs in Sight" Category :

Number of Data Points : 5,000 (for the Dog Face Model)

Reasoning :

This category comprises images where no dog faces are visible. With 5,000 images at hand in this category, the model gains exposure to scenes without dogs showcasing different settings, objects and backgrounds. By incorporating a number of examples the model is trained to differentiate between images that feature dog faces and those that do not thus minimising the chances of false positives. Training on a balanced dataset that includes both negative instances helps in avoiding biases and ensures that the model can accurately predict outcomes in various scenarios enhancing its overall performance.

Cat model (Cats present - No Cats present) :

Data Points : Over 9,000

Reasoning :

Like dogs, cats showcase facial characteristics based on breeds, colours and expressions. Hence a substantial dataset is essential to capture this variability and train a model proficient in detecting cat faces across contexts. Similar to dogs' diverse features, cats possess fur patterns, eye shapes and facial structures. When the model is trained on a dataset it can identify features, for detecting cat faces effectively making it more adaptable to different looks. Having plenty of images minimises the chance of favouring cat breeds or facial attributes. A varied dataset ensures that the model can recognize cat faces regardless of breed-related traits or poses.

Number of Categories : 2 (Presence of Cats, Absence of Cats)

Reasoning :

Similar to the dog model this centric approach employs classification to determine if an image includes a cat's face. With classes defined as "Presence of Cats" and "Absence of Cats " the model becomes adept at distinguishing between images containing cat faces and those that do not.
The binary classification method simplifies training procedures while enhancing evaluation processes. When the model focuses on telling apart images, with cats from those without it can effectively learn to recognize cat faces and make predictions on data.

"Presence of Cat(s)" Category :

Number of Data Points : 9,000 (for the Cat Face Model)

Reasoning :

Similar to the dog model this category represents images depicting cat faces. With 9,000 images allocated to this category the model can learn characteristics associated with cat faces such as breed variations, colours, expressions and lighting conditions. Cats display an array of features and poses akin to dogs. By training on a dataset, the model can effectively capture this diversity. Identify cat faces accurately under different circumstances. Including several images in this category aids in enhancing the model's ability to generalise while reducing the risk of overfitting to cat breeds or facial attributes.

"Absence of Cat(s)" Category :

Number of Data Points : 5,000 (, for the Cat Face Model)

Reasoning :

This category encompasses images where cat faces are not present.
Utilising 5,000 images, in this category offers the model an array of scenarios without cats encompassing backgrounds, objects and settings. By incorporating instances the model becomes adept at distinguishing between images featuring cat faces and those that do not thereby minimising alarms. Training on a dataset containing both positive and negative instances enhances the model's resilience. Ensures precise forecasts across different scenarios.

Human model (Human faces present - No Human faces present) :

Total Data Points : 7,200+

Reasoning :

Human facial characteristics can vary significantly due to factors like ethnicity, age, gender and facial expressions. Hence having a dataset is essential for training a model that can precisely spot faces across various demographics. Including several images assists the model in learning features for facial recognition, such as eye positions, nose shapes and mouth configurations. This diversity enables the model to generalise effectively to faces and environmental conditions.
Human faces may display poses, lighting situations and obstructions. By training on a dataset, the model can learn to identify faces in scenarios enhancing its performance, in real-world situations where such diversity is common. In essence, determining the quantity of data points for each model is influenced by the aim of encompassing an array of characteristics and differences found within each category (dogs, cats and humans). A broader and more varied dataset assists in reducing biases boosting performance and refining the model's capability to effectively categorise faces, in scenarios.

Number of Categories : 2 (Presence of Human Faces Absence of Human Faces)
Reasoning :

This model focuses on categorising images by determining whether human faces are present or absent. By creating two categories. In "Presence of Human Faces" and "Absence of Human Faces " the model's goal is to analyse images based on the existence of facial characteristics. Utilising a binary classification method simplifies the process. Enables the model to concentrate on distinguishing between images, with faces and those without. This clear separation allows the model to identify features, for detecting faces and make precise predictions.

"Presence of Human Face(s)" Category :

Quantity of Data Points : 7,200 (for the Human Face Model)

Reasoning :

This category signifies images where human faces are visible. The utilisation of 7,200 images in this group allows the model to grasp attributes associated with faces, including ethnicities, ages, genders, expressions and poses. Humans display a spectrum of features and emotions akin to animals. Training on empowers the model to capture this variability and accurately identify human faces under varying conditions. Having several images in this category enhances the model's adaptability and mitigates the risk of overfitting to demographic groups or facial traits.

"Absence of Human Face(s)" Category :

Quantity of Data Points : 5,000 (, for the Human Face Model)

Reasoning :

This category includes pictures that do not feature any faces. By utilising 5,000 images in this category the model is exposed to a range of scenarios of human faces encompassing various backgrounds, objects and settings. Incorporating instances, without faces aids the model in differentiating between images with faces and those without them consequently minimising incorrect identifications. Training on a rounded dataset that includes both positive and negative instances enhances the model's reliability. Guarantees precise forecasts, in diverse scenarios.

Here I'll share some methods to address data-related challenges, in machine learning tasks :

Data Augmentation :

Data augmentation techniques involve expanding the training dataset by applying transformations like rotation, flipping, cropping and scaling to existing images. These alterations introduce variations in the data enhancing the models adaptability to changes in orientation lighting conditions and object positions.

Data augmentation helps tackle issues stemming from a dataset size by boosting the diversity of available data. This approach can prevent overfitting. Enhance the models generalisation capacity particularly when training data is scarce or unevenly distributed across classes.

Transfer Learning :

Transfer learning entails utilising trained models – typically trained on extensive datasets – and fine tuning them on a smaller dataset specific to a particular domain. By transferring knowledge acquired from a source task to a target task transfer, learning can significantly reduce the dependency on labelled data and training duration.

Transfer learning proves valuable in addressing data related challenges when working with a dataset. By leveraging features learned from a dataset the model can deliver performance with less labelled data. This method is especially helpful, in situations where gathering an amount of labelled data's not feasible or cost effective.

Data Preprocessing :

Data cleaning entails identifying and fixing mistakes, discrepancies and unusual data points in the dataset. Preprocessing steps like normalisation, standardisation and feature scaling ensure that the data is formatted appropriately for model training. Additionally addressing missing values and handling class imbalances are elements of data preprocessing.

Having preprocessed data is essential for training machine learning models. By spotting and correcting errors in the dataset data cleaning enhances the quality and trustworthiness of the training data. Preprocessing methods guarantee that the data is properly scaled and normalised, facilitating model training and convergence.

Cross Validation :

Cross-validation involves dividing the dataset into subsets (folds) training the model on one subset and evaluating its performance on the remaining data. This process is repeated many times, with each fold acting as both a training set and a validation set. Cross-validation aids in evaluating the model's performance and its ability to generalise.

Cross-validation helps address issues related to variance. Prevent model overfitting.
By testing the model, on parts of the data cross validation offers a dependable assessment of its effectiveness and guarantees that the model can adapt well to new data. This method is crucial for identifying and resolving concerns like data leaks and biases, in the model.

Model Performance (20%-suggest: 700 words)

Human model (Human(s) present vs No Human(s) present) :

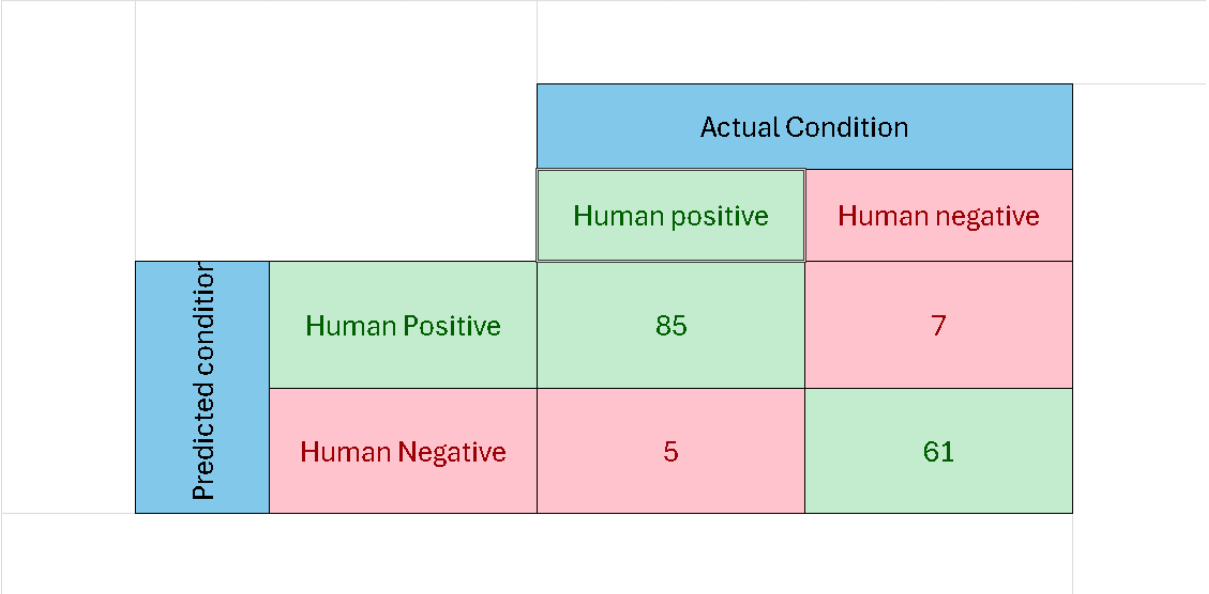


Figure 3 : Confusion matrix for the Human AI model (Human face present class)

■ Accuracy = (TP + TN) / (TP + TN + FP + FN)
= (85 + 61) / (85 + 61 + 7 + 5) = 0.924 or 92 %

This means that my AI model that is trained on Human faces is 92% accurate.

■ Precision = TP / (TP + FP)
= 85 / (85 + 7) = 0.923 or 92 %

This means that 92% of my predictions are positive out of all of the total positive predicted.

■ Recall = TP / (TP + FN)
= 85 / (85 + 5) = 0.9444 or 94 %

This means that my model is able to capture positive instances 94% of the time.

■ F1 Score = 2 * (Precision * Recall) / Precision + Recall)
= 2 * (0.92 * 0.94) / (0.92 + 0.94) = 0.929 or 93 %

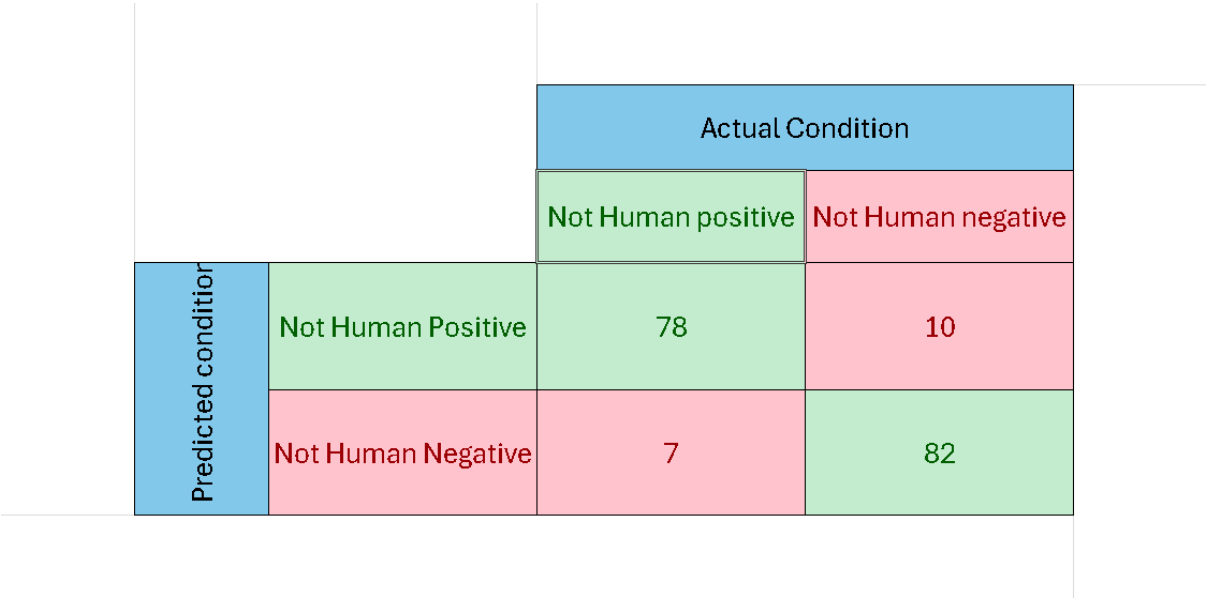


Figure 4 : Confusion matrix for the Human AI model (No Human face present class)

■ Accuracy = (TP + TN) / (TP + TN + FP + FN)
= (78 + 82) / (78 + 82 + 10 + 7) = 0.903 or 90 %

This means that my AI model that is trained on Human faces is 90% accurate.

■ Precision = TP / (TP + FP)
= 78 / (78 + 10) = 0.886 or 90 %

This means that 90% of my predictions are positive out of all of the total positive predicted.

■ Recall = TP / (TP + FN)
= 78 / (78 + 7) = 0.917 or 92 %

This means that my model is able to capture positive instances 92% of the time.

■ F1 Score = 2 * (Precision * Recall) / Precision + Recall)
= 2 * (0.90 * 0.92) / (0.90 + 0.92) = 0.909 or 91 %

Dog model (Dog(s) present vs No Dog(s) present) :

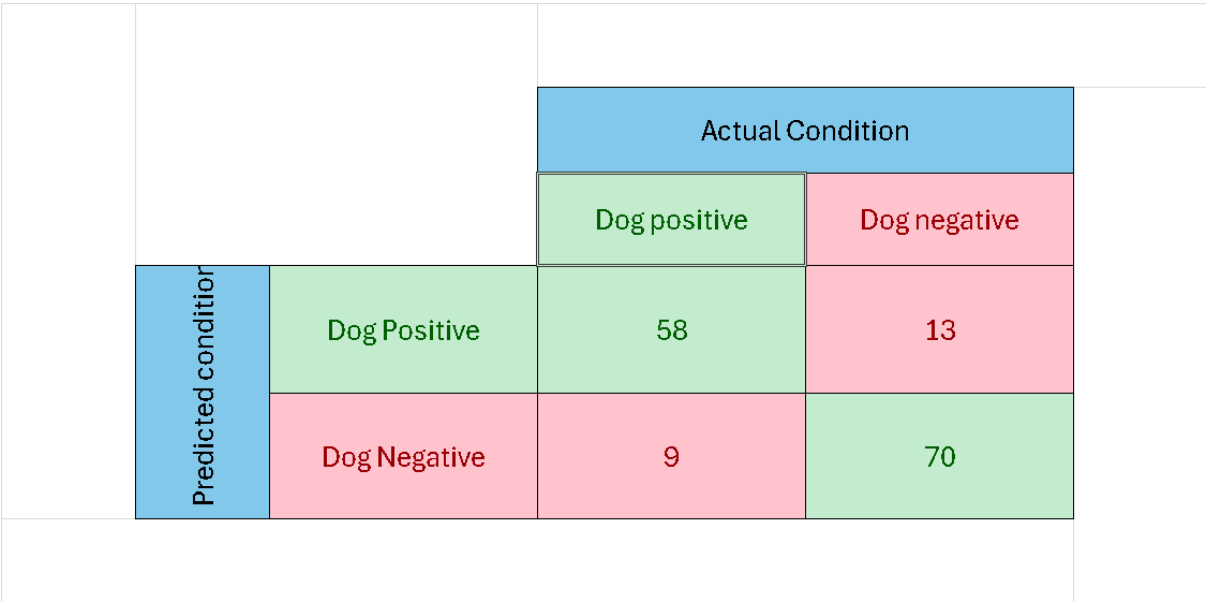


Figure 5 : Confusion matrix for the Dog AI model (Dog present class)

■ Accuracy = (TP + TN) / (TP + TN + FP + FN)
= (58 + 70) / (58 + 70 + 13 + 9) = 0.853 or 85 %

This means that my AI model that is trained on dog faces is 85 % accurate.

■ Precision = TP / (TP + FP)
= 58 / (58 + 13) = 0.816 or 82 %

This means that 82 % of my predictions are positive out of all of the total positive predicted.

■ Recall = TP / (TP + FN)
= 58 / (58 + 9) = 0.865 or 87 %

This means that my model is able to capture positive instances 87 % of the time.

■ F1 Score = 2 * (Precision * Recall) / Precision + Recall)
= 2 * (0.82 * 0.87) / (0.82 + 0.87) = 0.844 or 84 %

		Actual Condition	
		Not Dog positive	Not Dog negative
Predicted condition	Not Dog Positive	40	3
	Not Dog Negative	5	83

Figure 6 : Confusion matrix for the Dog AI model (No Dog present class)

■ Accuracy = (TP + TN) / (TP + TN + FP + FN)
= (40 + 83) / (40 + 83 + 3 + 5) = 0.938 or 94 %

This means that my AI model that is trained on dog faces is 94% accurate.

■ Precision = TP / (TP + FP)
= 40 / (40 + 3) = 0.930 or 93 %

This means that 93% of my predictions are positive out of all of the total positive predicted.

■ Recall = TP / (TP + FN)
= 40 / (40 + 5) = 0.888 or 90 %

This means that my model is able to capture positive instances 90% of the time.

■ F1 Score = 2 * (Precision * Recall) / Precision + Recall)
= 2 * (0.93 * 0.90) / (0.93 + 0.90) = 0.914 or 91 %

Cat model (Cat(s) present vs No Cat(s) present) :

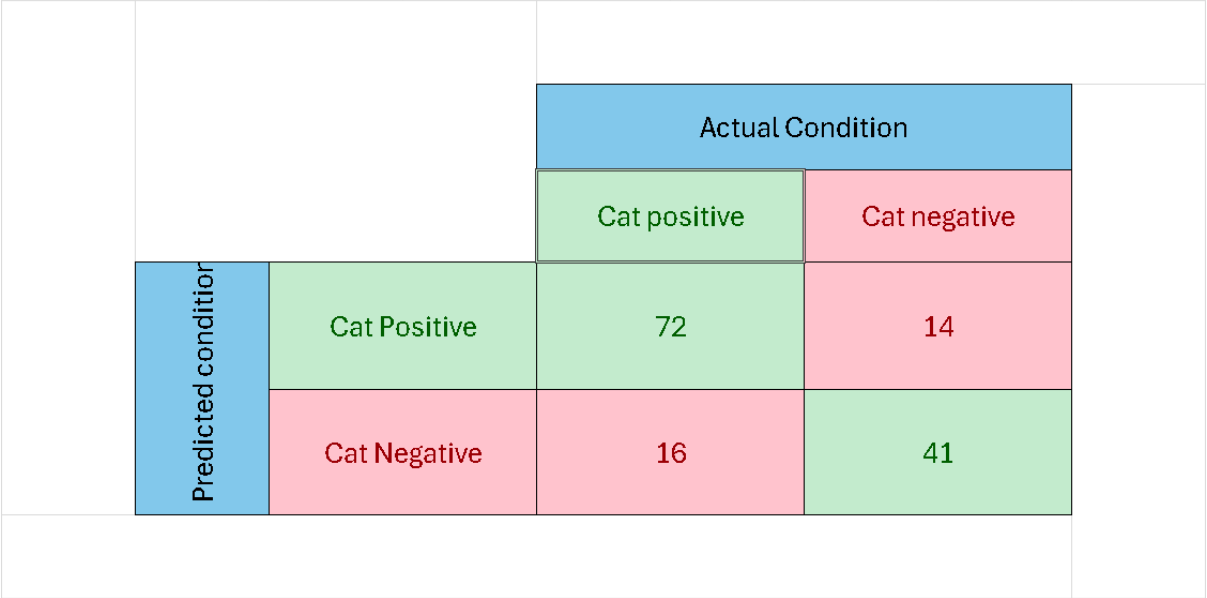


Figure 7 : Confusion matrix for the Cat AI model (Cat present class)

■ Accuracy = (TP + TN) / (TP + TN + FP + FN)
= (72 + 41) / (72 + 41 + 14 + 16) = 0.790 or 79 %

This means that my AI model that is trained on cat faces is 79 % accurate.

■ Precision = TP / (TP + FP)
= 72 / (72 + 14) = 0.837 or 84 %

This means that 84 % of my predictions are positive out of all of the total positive predicted.

■ Recall = TP / (TP + FN)
= 72 / (72 + 16) = 0.818 or 82 %

This means that my model is able to capture positive instances 82 % of the time.

■ F1 Score = 2 * (Precision * Recall) / Precision + Recall)
= 2 * (0.84 * 0.82) / (0.84 + 0.82) = 0.829 or 83 %

		Actual Condition	
		Not Cat positive	Not Cat negative
Predicted condition	Not Cat Positive	78	14
	Not Cat Negative	18	60

Figure 8 : Confusion matrix for the Cat AI model (No Cat present class)

■ Accuracy = (TP + TN) / (TP + TN + FP + FN)
= (78 + 60) / (78 + 60 + 14 + 18) = 0.811 or 81 %

This means that my AI model that is trained on cat faces is 81% accurate.

■ Precision = TP / (TP + FP)
= 78 / (78 + 14) = 0.847 or 85 %

This means that 85% of my predictions are positive out of all of the total positive predicted.

■ Recall = TP / (TP + FN)
= 78 / (78 + 18) = 0.812 or 81 %

This means that my model is able to capture positive instances 81% of the time.

■ F1 Score = 2 * (Precision * Recall) / Precision + Recall)
= 2 * (0.85 * 0.81) / (0.85 + 0.81) = 0.829 or 83 %

Critical discussion with reasoned explanations on different “edge cases” and “outliers” supported with two illustrated examples :

Edge Cases :

Descriptions :

- Uncommon Occurrences : Edge cases typically depict infrequent or atypical events that exist on the outskirts of the distribution. These occurrences might possess attributes or trends that challenge the model's capacity to generalise effectively.
- Uncertainty : In situations that are uncertain or vague where the boundaries between classes are blurred edge cases can emerge. These instances could display characteristics that make it challenging for the model to confidently categorise them potentially leading to mistakes or misclassifications.
- Out of Scope Data : Edge cases may also manifest when the model encounters data points that deviate from the distribution of the training data. These samples lying outside the range may exhibit unforeseen traits that were not part of the models training data making accurate predictions a difficult task.

Impacts on Models :

- Diminished Performance : Edge cases can result in reduced performance of machine learning models by questioning their assumptions and revealing vulnerabilities.
- Generalisation Challenges : Models struggling with edge cases may show generalisation performance by failing to capture the underlying patterns or subtleties in these instances.
- Error Risks : Mishandling edge cases can heighten error risks, in model predictions in applications where precise decision making is crucial.

Outliers :

Reasoned Explanations :

- Data Noise : Outliers can occur because of noise or errors during data collection resulting in data points that do not accurately represent the distribution.
- Genuine Anomalies : Outliers may also indicate anomalies or rare occurrences in the data, such as events or outliers that offer valuable insights.
- Influence on Models : Outliers can heavily impact model training by skewing the estimation of model parameters or guiding the learning process to accommodate these data points.

Impact on Models :

- Model Robustness : Outliers can challenge the resilience of machine learning models causing instability or bias during model training and inference.
- Overfitting : Models sensitive to outliers might overfit by adapting to these data points capturing noise instead of true patterns.
- Data Preprocessing Challenges : Effectively managing outliers often involves employing techniques, like outlier detection and removal transformations or using robust model structures less affected by outlier influence.

Empirical Evaluation (25% -suggest: 1300 words)

Average Accuracy found between my 6 classes (87%):

Accuracy, in its simplest form, represents the overall correctness of the model's predictions. An accuracy of 87% implies that the model correctly identifies the class labels for approximately 87% of the images in the dataset. While accuracy is a widely used metric for evaluating model performance, it may not always provide a complete picture, especially in scenarios where the dataset is imbalanced. In such cases, a high accuracy score can be misleading if the model predominantly predicts the majority class, potentially masking significant misclassifications in minority classes. Therefore, while an accuracy score of 87% is respectable, I need to conduct further analysis to understand how well the model performs across different classes and whether it exhibits any biases or limitations.

Average Precision found between my 6 classes (88%):

Precision measures the ratio of true positive predictions to the total number of positive predictions made by the model. A precision score of 88% means that when the model predicts a particular class (e.g., “No Human face(s) Present” or “No Dog(s) Present”), it is correct approximately 88% of the time. Precision is particularly valuable in scenarios where false positives carry significant consequences or costs. However, a high precision score does not necessarily guarantee a good model, as it may come at the expense of recall. In simple terms, a model with high precision may be overly conservative in making positive predictions, leading to a lower recall score.

Average Recall found between my 6 classes (88%):

Recall (which is also known as sensitivity) quantifies the model's ability to identify all relevant instances of a particular class in the dataset. An 88% recall score implies that the model correctly identifies approximately 88% of all instances belonging to a specific class. Recall is particularly critical in scenarios where false negatives (which is missed detections) are costly or undesirable. However, similar to precision, a high recall score may accompany low precision if the model tends to make many positive predictions, including false positives. Therefore, achieving a balance between precision and recall is essential for a well-performing model.

Average F1 Score found between my 6 classes (88%):

The F1 score, which is the mean of precision and recall, provides a single metric that balances the trade-off between precision and recall. An F1 score of 88% indicates that the model achieves a harmonious balance between minimising false positives and false negatives. The F1 score is particularly useful when dealing with imbalanced datasets, where precision and recall may vary significantly across different classes. However, it's important to note that the F1 score might not fully capture the nuances of a model's performance, especially in scenarios where precision and recall have different importance.

Overall Evaluation:

Strengths:

The model's reliability and robustness are evident, through its performance on evaluation metrics when classifying images. The balanced precision, recall and F1 score highlight the models ability to strike a balance between reducing positives and false negatives which is vital, for practical use cases. The reliability and robustness of these models comes from the combination of utiliting thousands of images for training which gives the model the diversity of samples to actually learn effectively and the decision to train these models at 100 epochs as opposed to the default which is 50, having double the epochs just means the models will know their sample that much more and when diversity is considered, this can only improve reliability.

Weaknesses:

Although the model performed well overall, it would be beneficial to conduct a detailed examination to assess how it handles individual categories and to uncover any biases or constraints that may exist. Delving into the errors can reveal themes, in misclassifications and highlight areas needing enhancement where the model falls short.Highlighting these areas will allow me to continuously improve the reliability of the model's outcomes.

Going further ahead :

To truly grasp the reasons behind misclassifications delve into an error analysis. I could look into ways to tackle them like broadening the dataset or refining the model's structure. Keeping an eye on how the model performs with new data and adjusting the training process accordingly to handle any emerging challenges or shifts in data patterns, Thinking about using techniques or stacking models to boost performance and reduce the risks of overfitting or bias. In summary, even though the AI agent and model show results, it is crucial to conduct a thorough assessment that identifies specific strengths, weaknesses and areas for improvement to optimise their effectiveness, in real world image classification tasks.

Potential Anomalies and detected issues :

Performance Disparities Among Categories :

One issue that may occur is when certain categories are disproportionately represented in the dataset. For instance if the dataset mostly consists of pictures of a type of dog or cat the model might become biassed towards recognizing those breeds accurately while struggling with less common breeds. Likewise biases can be present in data like an overrepresentation of particular demographics (such as gender or ethnicity) leading to skewed model performance.

The possible causes of biases in data representation can stem from factors like sampling during data collection, societal biases reflected in the dataset or limitations in diversity and coverage.

A potential way to improve this is to expand and diversify the dataset : Include images that cover a range of breeds, species and demographic groups to reduce biases and ensure representation.Utilise data preprocessing methods : Employ techniques like balancing data or using sampling to distribute samples across various categories minimising the impact of imbalances on model training.

Conduct bias assessments : Regularly assess the dataset for biases using measures, like parity or equalised odds to identify and address biases.

Performance Disparities Among Categories :

Another irregularity may appear in performance variations, among categories. For example the model might excel in distinguishing types of dogs or cats while facing challenges with others. Similarly it could show differing levels of precision in recognizing faces across demographics.

Performance differences among categories could stem from distinctions in the characteristics and complexities of each category. Some categories may have attributes or present difficulties that were not adequately addressed during model training.

A potential way to improve this would be to fine tune models for categories : Adjust the models design or tweak parameters to tackle issues to poorly performing categories like incorporating specialised loss functions or increasing model capacity.Utilising transfer learning from tasks, Make use of trained models or transfer learning methods to apply knowledge from tasks with comparable categories aiding the model in better adapting to challenging classes. Merge predictions from models or ensemble strategies like bagging or boosting to alleviate performance variations and enhance overall classification accuracy across different categories.

Ethical and Social Considerations :

Challenges and concerns could also emerge from social aspects, such as biases reflecting norms or ethical dilemmas concerning privacy and consent, in collecting image data.

Biases and ethical concerns in machine learning models may arise from inequalities, societal norms or systemic injustices ingrained in the data. Neglecting these issues could. Exacerbate existing biases resulting in outcomes and moral quandaries.

A way to improve this could be to embrace ethical frameworks and standards like fairness, accountability and transparency (FAT) principles to steer the development and implementation of models ensuring fairness, transparency and accountability.Diverse Teams with Inclusion : Cultivate inclusive teams comprising researchers and practitioners with backgrounds to offer diverse viewpoints and address ethical and societal aspects effectively.Engagement with Stakeholders : Involve stakeholders such as impacted communities and field experts, at every stage of model development to grasp perspectives pinpoint biases and collectively devise solutions that prioritise fairness and equality.

Central neural networks :

Central pattern generators (CPGs) also known as networks are neural circuits present in animals' central nervous systems (CNS) that create rhythmic patterns of motor activity like walking, swimming or breathing autonomously without relying on sensory input. These networks play a role in producing and synchronising movements and are present across various organisms ranging from simple invertebrates to mammals. There are characteristics and principles that define neural networks :

- Rhythmicity : CPGs generate patterns of activity crucial for producing rhythmic movements such as walking or breathing.
- Modularity : Central neural networks consist of components that can be activated or inhibited to produce activity patterns. This modular structure provides flexibility in generating movements.
- Feedback Control : While CPGs can create activity independently of feedback they can also incorporate sensory information to adjust the rhythmic patterns of activity based on environmental changes or internal conditions.
- Hierarchical Organisation : In organisms central neural networks exhibit organisation where higher level circuits regulate the functioning of lower level circuits. This hierarchical arrangement enables the coordination of movements involving body parts.
- Neural Plasticity : The systems neural networks have the ability to adapt and change their structure and function based on experiences or injuries. This adaptability plays a role in learning, adjusting to environments and recovering from neural damage.
- Evolutionary Consistency : While the specific organisation of networks may differ among species there are fundamental principles that remain consistent throughout evolution. This indicates that these networks play roles in coordinating movements across a wide range of organisms.
- Extensive Research : Scientists have extensively studied central neural networks in various model organisms including invertebrates like leeches and lobsters as well as vertebrates such as lampreys and rodents. These studies have offered insights into how rhythmic movements are generated and controlled in animals.

The Structure of Central Neural Networks :

Neuronal Connectivity : Central neural networks are composed of interconnected neurons within the system (CNS). These networks can range from circuits found in invertebrates to complex structures in vertebrates.

Segmented Arrangement : In species central neural networks display a segmented arrangement where each body segment has its set of neural circuits responsible for coordinating rhythmic movements. This segmented organisation aids in movement, across body parts.

Connections Between Neurons :

The connection between neurons in networks is crucial for creating rhythmic activity patterns. Interneurons play a role in facilitating interactions among inputs, motor outputs and the inherent rhythmic activity.

Central Neural Networks Functions :

Generating Rhythmic Patterns : Central neural networks primarily function to create activity patterns that control rhythmic movements like walking, swimming or breathing. These networks can produce coordinated motor output without feedback showcasing their inherent rhythm.

Influenced by Sensory Feedback : While central neural networks can independently generate activity they are also influenced by feedback. Inputs from proprioceptors, mechanoreceptors and other sensory organs can impact the frequency, strength and timing of movements enabling organisms to adjust to changing environments.

Managing Complex Movements : Central neural networks coordinate movements involving body parts and muscle groups. By harmonising the actions of motor neurons and muscle fibres these networks ensure effective performance of motor tasks.

Importance of Central Neural Networks :

Central neural networks are vital for animal survival and reproduction.Central neural networks play a role in behaviours like moving, eating and reproducing. Any disruptions in these networks can significantly impact an organism's ability to function and engage with its surroundings.

Central neural networks act as models for studying how neural circuits are organised, change and work. By figuring out how these networks produce and regulate movements researchers can learn more about the principles of how the brain processes information and influences behaviour.

Insights from studying networks can help in developing treatments for neurological conditions. Understanding how these networks react to damage or illness can guide the creation of strategies for repairing and rehabilitating nerves.

The design of systems that can move independently has been influenced by central neural networks. By imitating the principles that govern neural networks researchers are working on creating robots capable of moving swiftly and adapting to complex environments.

In conclusion, exploring networks is an exciting area of study with implications for understanding how neural circuits are structured and operate as well as for developing treatments and bio-inspired robotics. Further exploration into the organisation, operation and adaptability of these networks holds promise for revealing insights into the intricacies of brain function and behaviour.

Types of Central Neural Networks :

- **Spinal Cord Networks :** In vertebrates central neural networks within the cord manage movements, like walking and swimming. These networks consist of interneurons and motor neurons organised into pattern producing circuits known as pattern generators (CPGs).
- **Brainstem Networks :** The brainstem houses networks responsible for regulating essential bodily functions like breathing, heartbeat and swallowing. These networks produce rhythmic activity patterns for sustaining life.
- **Networks in the Midbrain and Forebrain :** motor behaviours, such as reaching, grasping and vocalisation are controlled by central neural networks in higher brain regions like the midbrain and forebrain. These networks combine input with processes to coordinate movements effectively.

Operational Mechanisms :

Reciprocal Inhibition : A common mechanism found in networks involves reciprocal inhibition, where alternating activation and inhibition of opposing muscles or motor neurons generate rhythmic activity patterns. This mechanism helps synchronise limb flexion and extension during movement.

Synaptic Plasticity : Within networks synaptic plasticity allows for adjustments in synaptic strength and connectivity based on activity patterns or sensory input. This adaptability supports learning, memory formation and network adaptation.

Neuromodulation : Neuromodulatory signals such as neurotransmitters and hormones can dynamically influence the activity of networks by modifying their output patterns in response to changes in internal conditions or external factors.

Applications of Central Neural Networks :

Neurorehabilitation : Insights into the functioning and adaptability of networks can guide the development of rehabilitation approaches for individuals with neurological injuries or disorders. Interventions targeted at areas can help in repairing functions and restoring motor skills by leveraging the adaptability of central neural networks.

Neuromorphic Engineering involves drawing inspiration from networks to design artificial neural networks and robotic systems capable of independent and flexible behaviour. Engineers strive to create robots, with improved movement abilities and motor proficiency by mimicking how CNNs operate.

Control Systems utilise networks to develop autonomous vehicle control systems, drones and other robotic platforms. By replicating the modular structure of CNNs these control systems can better adjust to ever-changing environments.

Below I will include links to my teachable machine models:

Dogs or not dogs classification:
Model - https://teachablemachine.withgoogle.com/models/sxCA-62hq/

Cats or no cats classification:
Model - https://teachablemachine.withgoogle.com/models/p1NR5Kpxv/

Human or no Human classification:
Model - https://teachablemachine.withgoogle.com/models/Gq5OvKizL/

Appendix - Demonstration Evidence (30%)

Examples of True Positives for each classifier:

"Human faces" :

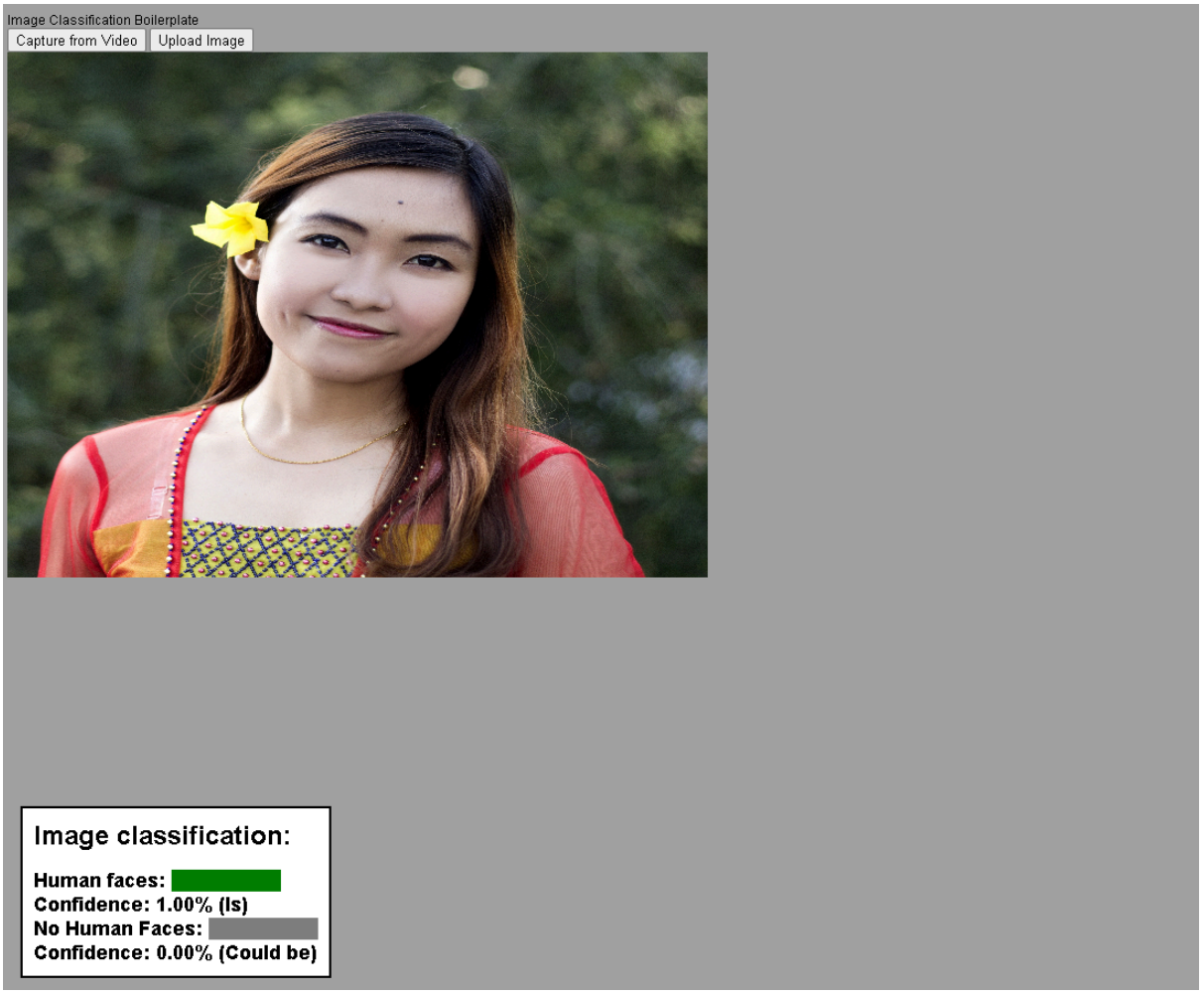


Figure 9 : True positive example for "Human faces" classification

“Dog(s) Present” :

Figure 10 : True positive example for “Dog(s) Present” classification

“Cat(s) Present” :

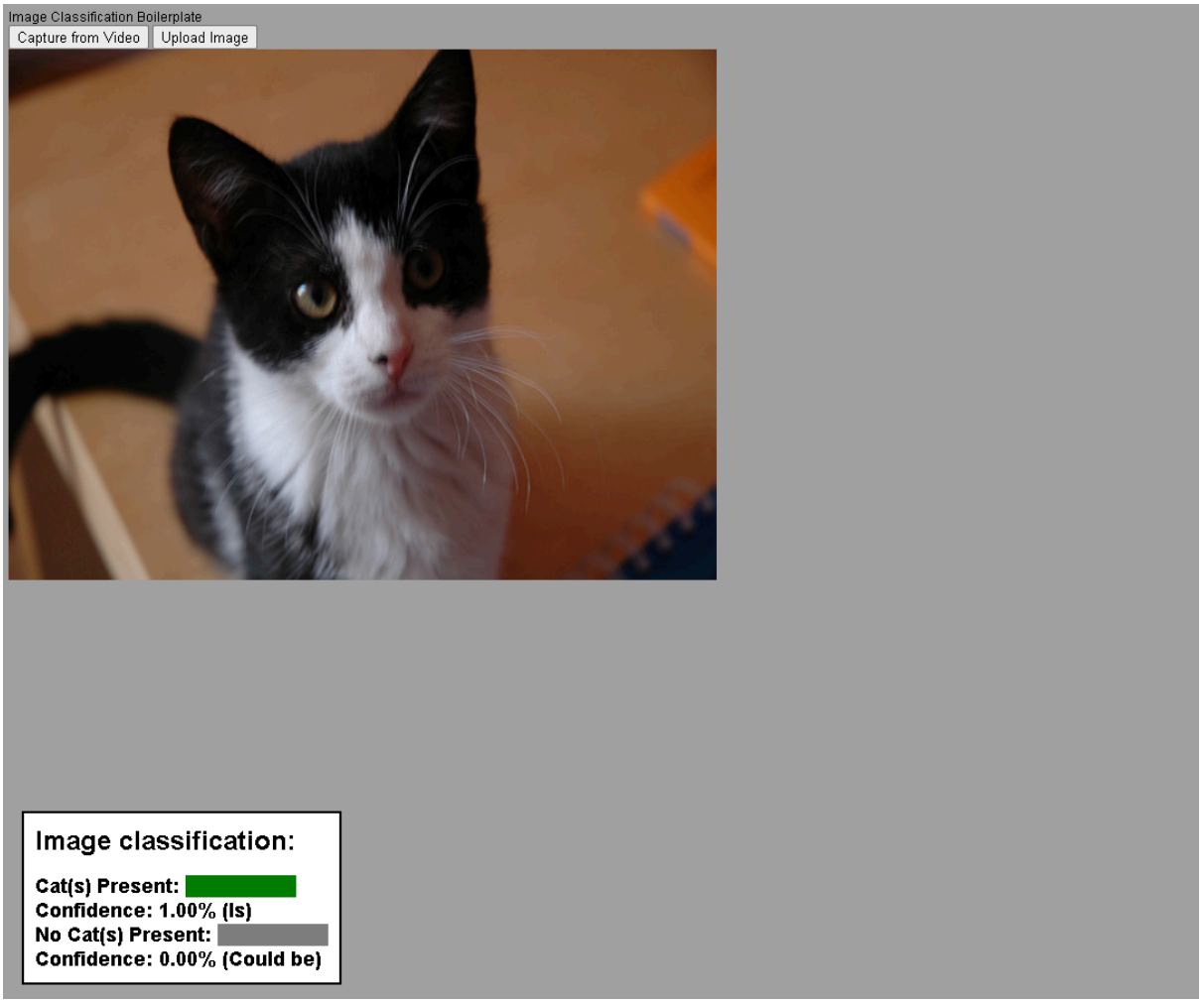


Figure 11 : True positive example for “Cat(s) Present” classification

Examples of True Negatives for each classifier:

“No Human Faces” :

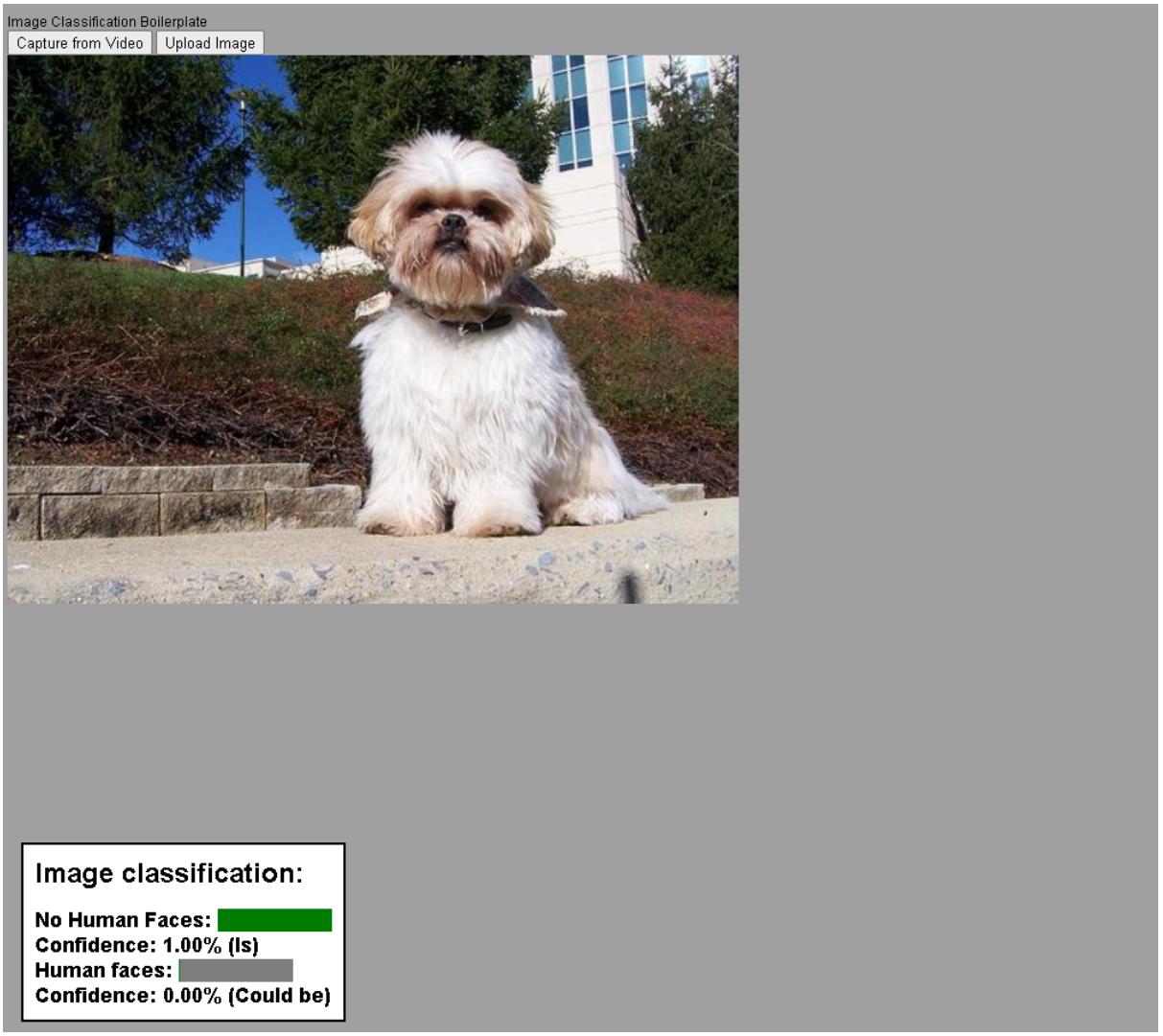


Figure 12 : True Negative example for “No Human faces” classification

“No Dog(s) Present” :

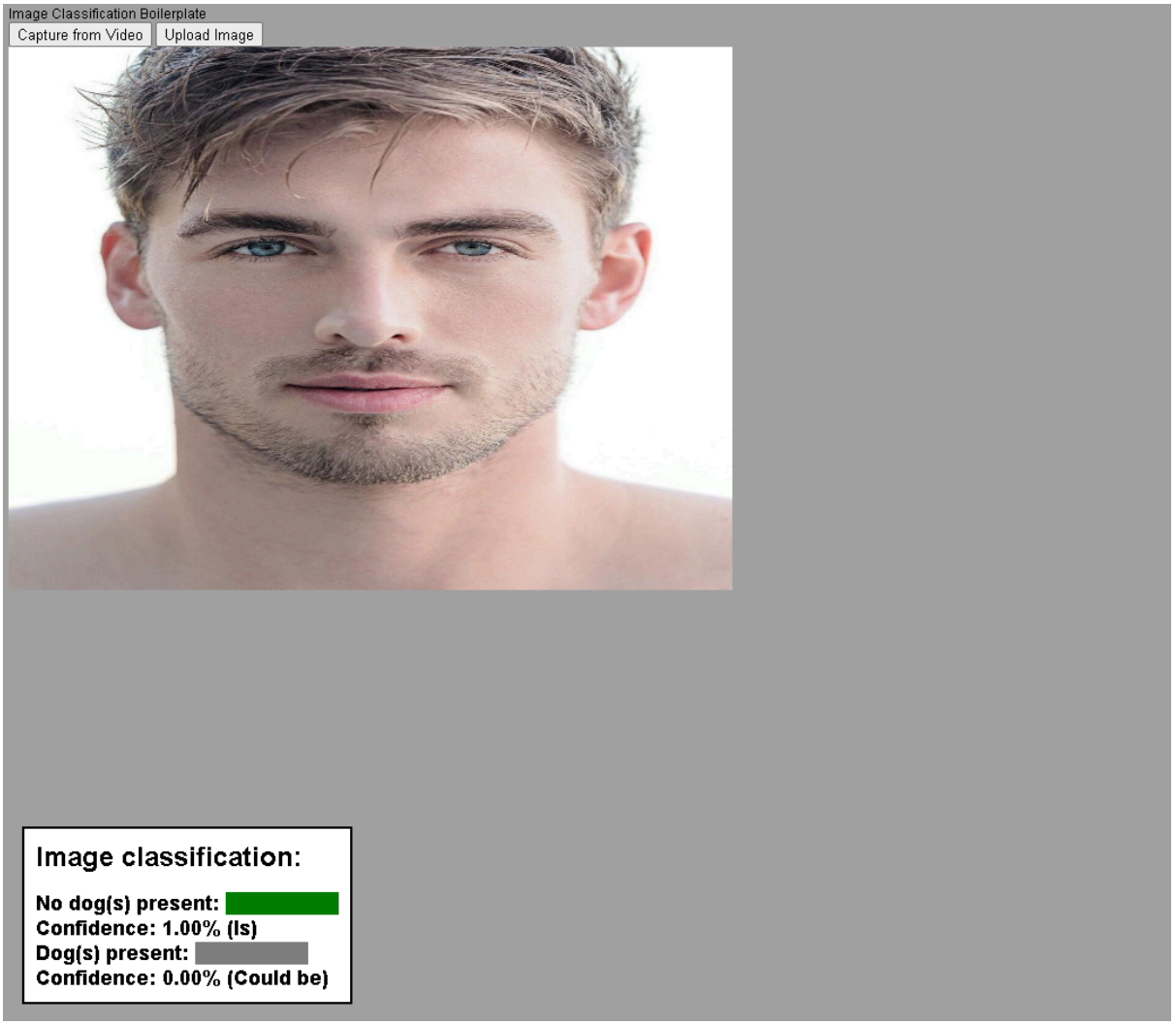


Figure 13 : True Negative example for “No Dog(s) Present” classification

“No Cat(s) Present” :

Figure 14 : True Negative example for “No Cat(s) Present” classification

Differentiated examples of FNs supported with clear and critically considered annotations explaining why it was likely misclassified :

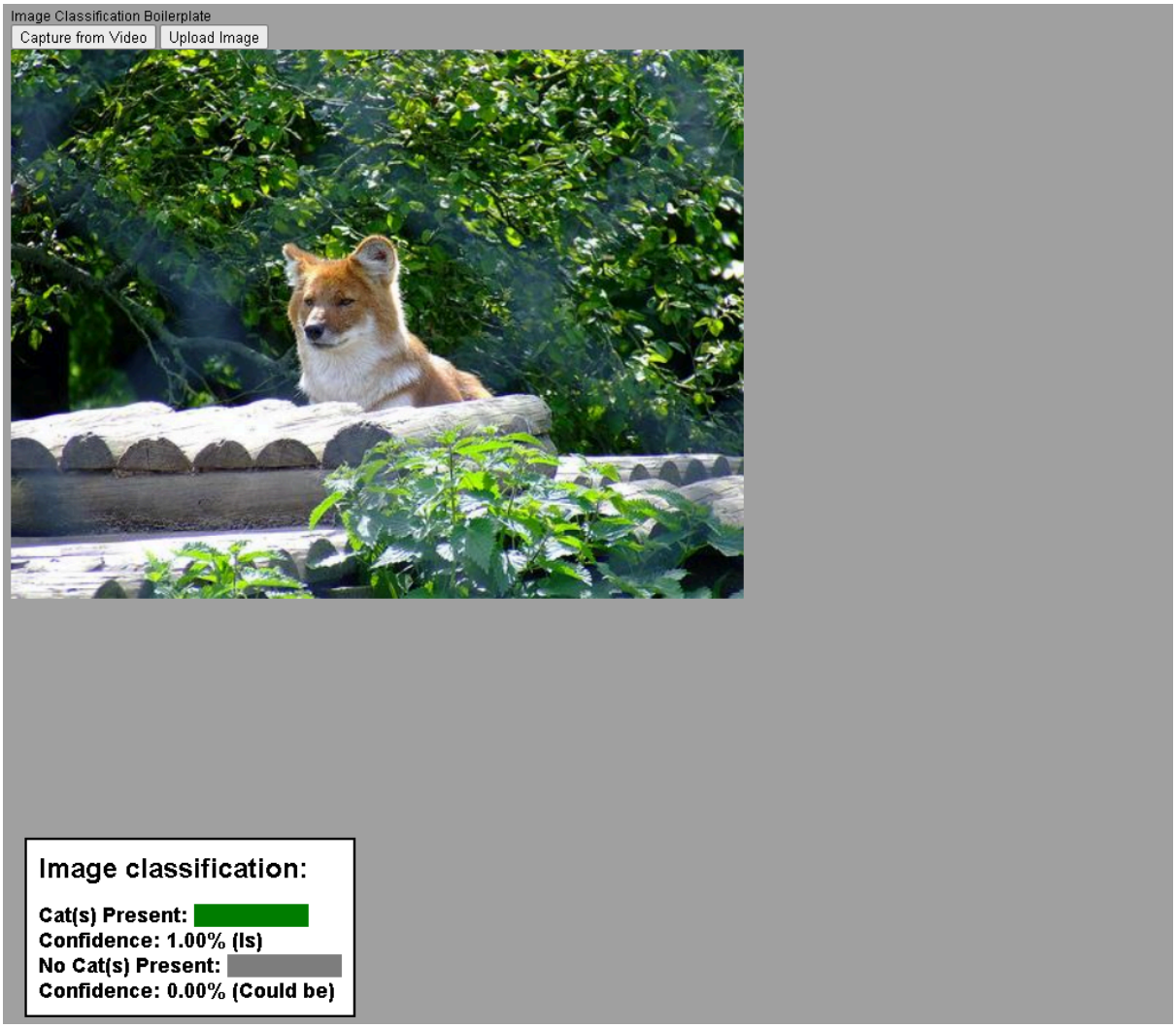


Figure 15 : False Positive example for “Cat(s) Present” classification

This is a False Positive as the condition of cats being present is absent and the image has been misclassified as a cat when the image provided is clearly a dog. The reason I believe this image may have been misclassified is because in this particular image, the dog looks quite similar to a cat with some of its features, for example, you can see that the dogs fur is quite visible which is a feature that cats share, the ear shape, although very pronounced and to the human eye, is clear to be the ears of a dog, to the model it will also look very similar to the ear of a cat. These common features are the main reason I believe that the image was misclassified. Another possible reason why the image was misclassified could be that the Cat model was trained on more images of a cat, creating a training data bias where the model has a bias towards classifying images as there being cats present. Another possible reason for this misclassification is the AI might have been trained on a limited set of images that didn't fully represent the diversity of dog breeds, leading it to incorrectly classify certain dogs as cats because it hasn't learned to recognize their specific features.

Figure 16 : False Positive example for “Human Faces” classification

This is a False Positive as the condition of Human face being present is absent and the image has been misclassified as a human face being present when the image provided is clearly a kitten.The reason I believe this image has been misclassified may be a result of the hand holding the kitten up, this could be because the model has been trained on images with people where their hands and arms are visible so the model may have associated this with human faces being present and therefore it has misclassified this image as human faces being present.

Differentiated examples of FPs supported with clear and critically considered annotations explaining why it was likely misclassified :

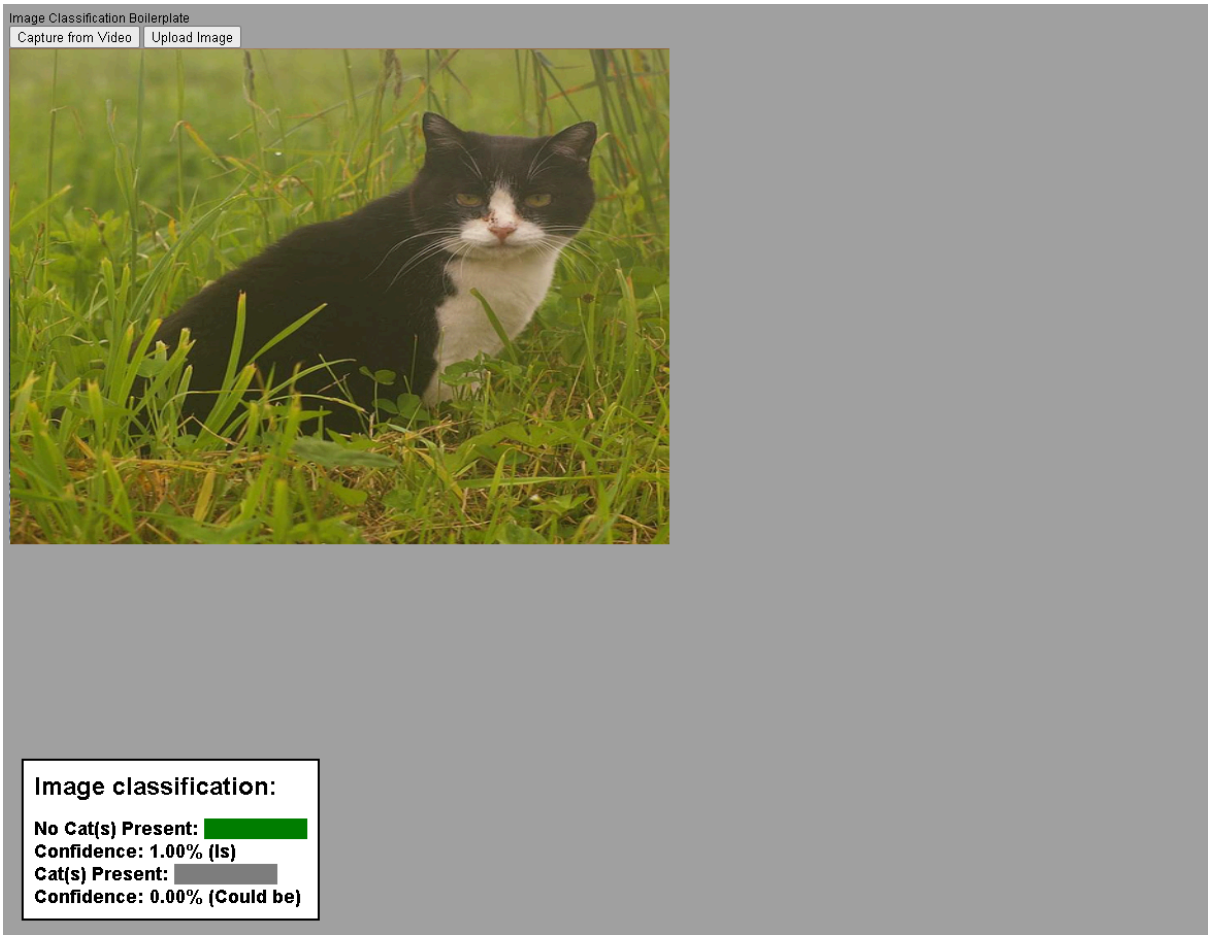


Figure 17 : False Negative example for “No Cat(s) Present” classification

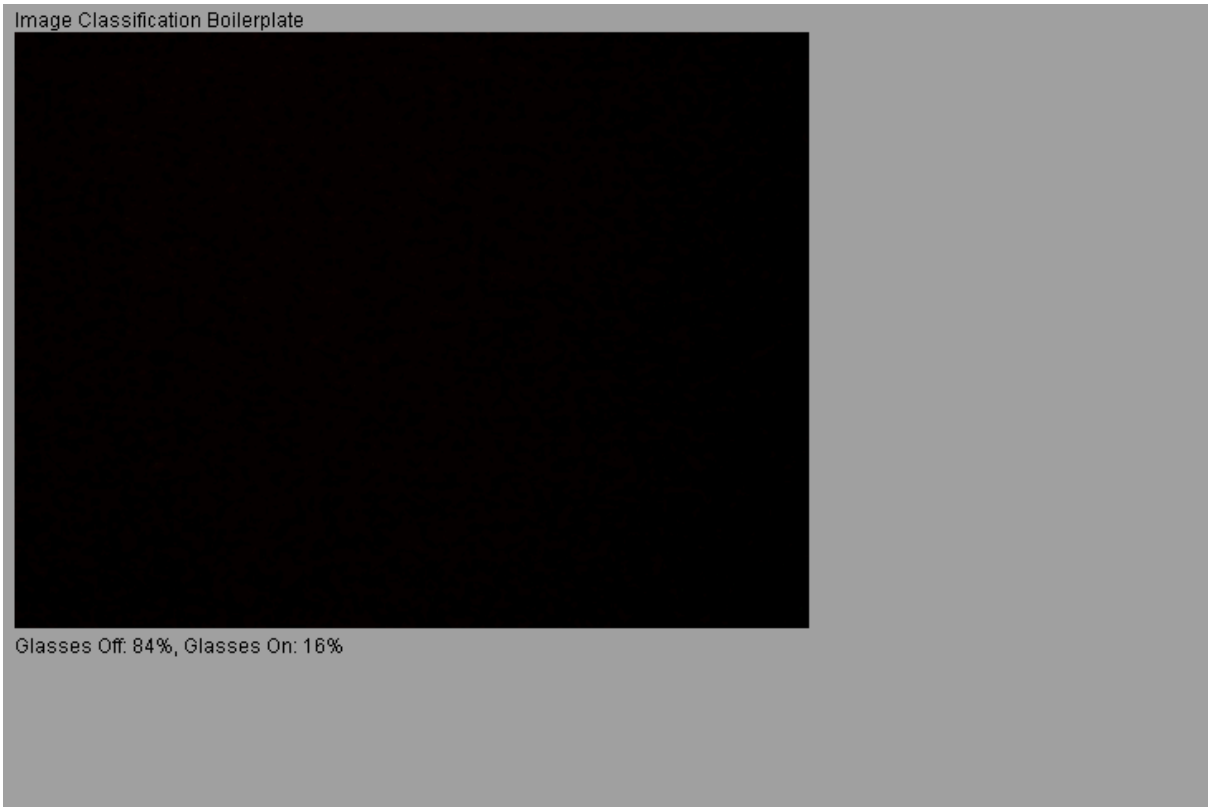
This is a False Negative as the condition of the cat being present is not absent but the image has been classified as there being no cat present in the image, One reason for this image being misclassified could be that certain types of cats those, with fur patterns or facial characteristics could look similar to certain breeds of dogs. If the model’s training dataset did not include enough of a range of examples, for both cats and dogs it may struggle to distinguish between these visual similarities.Another potential reason for misclassification is that the picture isn't clear or the cats characteristics are hard to see.Finally another reason for this misclassification could be mistakes made during the image processing stage like adjusting size, cropping or standardising may inadvertently alter the cats features to appear dog like in the eyes of the model.

Figure 18 : False Negative example for “No Dog(s) Present” classification

This is a False Negative as the condition of the dog being present is not absent but the image has been classified as there being no dog present in the image, One reason that this image may have been misclassified is that the dog model may have not been trained on this breed of dog so it may have gotten confused and classified this image as there being no dogs present, Another reason why this may have been misclassified is the colourful clothing and accessories that the dog is wearing in the picture, the model may not expect this but in reality a lot of people put clothes on their dogs, if the model has not been trained to view and classify pictures of dogs with clothes then this can also confuse the model, causing it to misclassify the image.

Three differentiated examples illustrating the expected and formatted agent outputs :

Expected agent output:



Formatted Agent Output :

In the formatted Agent output you can see the enlargement of the Image being classified which can make it easier to see, there is also the function of uploading an image which the expected output does not have, and finally the formatted agent output has a image classification section at the bottom left of the output, these classifications have been placed on a white background to make it more visible and confidence bars have been added, giving this an aspect of visualisation for the confidence levels.

Screenshots illustrating your Agent's code blocks responsible to interpreting and formatting the results from your model :

```
"use strict";

var imageModelURL = ' https://teachablemachine.withgoogle.com/models/Gq5OvKizL/';
var classifier;
var cam;
var uploadedImage;
var label0 = "", confidence0 = 0;
var label1 = "", confidence1 = 0;

function preload() {
  classifier = ml5.imageClassifier(imageModelURL + 'model.json');
}

function setup() {
  var viewport = createCanvas(640, 480);
  viewport.parent('video_container');
  frameRate(24);

  cam = createCapture(VIDEO, function() {
    classify();
  });
  cam.hide();

  var resultsContainer = document.getElementById('results');
  resultsContainer.style.position = 'absolute';
  resultsContainer.style.bottom = '20px';
  resultsContainer.style.left = '20px';
  resultsContainer.style.backgroundColor = 'white';
  resultsContainer.style.padding = '10px';
  resultsContainer.style.border = '2px solid black';

  var fileInput = document.getElementById('fileInput');
  fileInput.addEventListener('change', handleFile);
}

function classify() {
  if (uploadedImage) {
    classifier.classify(uploadedImage, processResults);
  } else {
    classifier.classify(cam, processResults);
  }
}

```

Figure 17 : Code block responsible for interpreting and formatting model results (Part 1)

preload(): This function is part of the p5.js library and is automatically called before the setup() function. In this code, it's used to load the machine learning model using the ml5.js library. The imageClassifier function loads the model's JSON file from the specified URL.

setup(): This function is also part of the p5.js library and is called once when the program starts. It sets up the canvas where the video feed and results will be displayed. It also initialises the webcam using createCapture() and hides the default DOM element created by p5.js for the webcam. Additionally, it sets up the DOM elements for displaying results and for handling file input.

classify(): This function is responsible for classifying the uploaded image or webcam feed using the loaded model. It calls the classifier.classify() function from ml5.js library, passing either the uploaded image or the webcam feed as parameters.

```
function processResults(error, results) {
  if (error) {
    console.error("Classifier error: " + error);
  } else {
    label0 = results[0].label;
    confidence0 = results[0].confidence;
    label1 = results[1].label;
    confidence1 = results[1].confidence;
  }
}

function friendlyResults() {
  let result = "<span style='font-size:24px; font-weight:bold;'>Image classification:</span><br><br>";
  if (label0.length > 0) {
    let label0Classification = getConfidenceClassification(confidence0);
    let label1Classification = getConfidenceClassification(confidence1);

    let progressBar0 = createProgressBar(confidence0);
    let progressBar1 = createProgressBar(confidence1);

    result += "<span style='font-size:18px; font-weight:bold;'>" + label0 + ": " + progressBar0 + "<br> Confidence: "
  }
  return result;
}

function getConfidenceClassification(confidence) {
  if (confidence <= 0.4) {
    return "Could be";
  } else if (confidence > 0.4 && confidence <= 0.75) {
    return "Likely";
  } else {
    return "Is";
  }
}
```

Figure 18 : Code block responsible for interpreting and formatting model results (Part 2)

processResults(error, results): This function is a callback that is executed after the classification process is complete. It receives the results of the classification, including labels and confidence scores. It updates global variables label0, confidence0, label1, and confidence1 with the results.

friendlyResults(): This function generates a human-readable representation of the classification results. It formats the labels, confidence scores, and confidence classifications (could be, likely, is) into HTML elements for display.

getConfidenceClassification(confidence): This function categorises the confidence scores into three classes: "Could be", "Likely", or "Is", based on predefined thresholds. It returns the appropriate classification based on the confidence score provided as input.

```
function createProgressBar(confidence) {
  let color;
  if (confidence <= 0.4) {
    color = "red";
  } else if (confidence > 0.4 && confidence <= 0.75) {
    color = "yellow";
  } else {
    color = "green";
  }
  return "<progress value='" + (confidence * 100) + "' max='100' style='width:100px; height:20px; background-color: #f0f0f0;' title='"
}

function draw() {
  background("#c0c0c0");

  if (uploadedImage) {
    image(uploadedImage, 0, 0, width, height);
  } else {
    image(cam, 0, 0, width, height);
  }

  document.getElementById("results").innerHTML = friendlyResults();

  classify();
}

function handleFile() {
  var file = this.files[0];
  uploadedImage = loadImage(URL.createObjectURL(file), function(img) {
    uploadedImage = img;
    cam.stop();
    classify();
  });
}
```

Figure 19 : Code block responsible for interpreting and formatting model results (Part 3)

createProgressBar(confidence): This function creates an HTML progress bar element based on the confidence score. It assigns a colour to the progress bar based on predefined confidence thresholds and sets the value and maximum attributes accordingly.

draw(): This function is part of the p5.js library and is called continuously. It redraws the canvas and updates the display with the current webcam feed or uploaded image. It also updates the results container with the latest classification results by calling friendlyResults() and triggers the classification process again using classify().

handleFile(): This function is called when a file is selected using the file input element. It loads the selected image file, converts it into an image object, stops the webcam feed, updates the global variable uploadedImage, and triggers the classification process.

References:

Bansall, S. (2019). *Agents in Artificial Intelligence - GeeksforGeeks*. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/agents-artificial-intelligence/.