Deep Learning for Human and Animal Face Recognition using CNNs

Kareem Piper, Sara Parrish, and Hooman Sabarou

Department of Mathematics & Statistics, University of West Florida

IDC6146: Deep Learning

Dr. Shusen Pu

February 28, 2025

Supplementary Links

Google Drive (Final Code and Report):
https://drive.google.com/drive/folders/1_RiAiZTgRalsb7JB36oFdaFyRmzlgPpB?usp=drive_link

Presentation:
https://drive.google.com/file/d/1Lq92z1WTV4ncCfd_27o1xo8kI_FhX9Jx/view?usp=drive_link

Statement on Project Contributions

This project's concept and data sourcing as well as initial notebook was conceived and produced by Kareem Piper. Once there was group consensus (i.e., that we build a novel project), Kareem created and shared the initial team google drive which contained a zip folder with the project's training sets, test sets and validation sets, data splitting and labeling code, as well as the initial project notebook code. That google drive was then refined by Sara Parrish who then re-shared it with the team so that the team could begin to formally build out the project together. During the tenure of this project the team communicated daily via Discord and met once per week via Zoom to discuss with each other what they did concerning the project as well as to share feedback. Below is a concise breakdown of what each team member did on a weekly basis all the way up to final project submission:

Week 1 (Jan 27th-30th): Project Proposal and Initial Proposal Submission

- Kareem: Proposed project to teammates, wrote and submitted "Project 1 Proposal"

Proposal: 1/29

Week 2 (Feb 3rd -Feb 7th): Code Completion & Initial Paper Draft

- Kareem: Sourced and preprocessed project data finalized initial model and presented initial results.
- Hooman: Debug and optimize training performance.
- Sara: Created initial paper outline and draft introduction and submitted our first progress report.
- Team: Reviewed and refined the draft.

Week 3 (Feb 10th-Feb 14th): Feedback Edits, Additional Visualizations, Refined Drafts

- Kareem: Implemented edits to project code based on instructional staff's feedback.
- Hooman: Expanded and refined paper sections and submitted our second progress report.
- Sara: Generate additional visualizations (confusion matrix, performance graphs).
- Team: Finalized conclusions, proof the document.

Week 4 (Feb 17th-21st): Hyper Parameter Tuning/Training, Working on Final Paper

- Kareem: Provided team detailed notes on his training experimentation and outputs. Also worked on the Advanced Data Processing and Advanced Mathematical Concepts section of the paper.
- Sara: Filled out final report through Challenges, save and add visualizations to report (made ROC, scatter, misclassifications). Experimented with hyperparameter optimization with halving grid search. Looked into a ResNet model for comparison.
- Hooman: Reviewing codes, documenting model architecture, training configurations. Writing, reviewing, and editing the final report (results, challenges, potential applications, and conclusions).
- Team: Continued to finalize the final notebook and proof the final paper.

Week 5 (Feb 24th-28st): Refining Final Project Notebook, Paper, and Google Drive Project Submission

- Kareem: Added Appendices in paper, edited paper, worked on cleaning up drive, and final notebook
- Hooman: Explored real world applications, enhanced and extended literature review, reviewed the report and codes, reviewed impacts, and conducted citation analyses.
- Sara: Conducted final edits on paper to ensure APA compliance. Constructed initial deck for presentation.
- Team: Built and recorded Project Presentation

Deep Learning for Human and Animal Face Recognition using CNNs

Image classification is a crucial application of deep learning, leveraged for use in a variety of applications for humans such as Automatic Facial Recognition (AFR) in biometric authentication and surveillance, Facial Expression Recognition (FER) to be used for diagnosis of disease (Hallowell et al., 2023), as well as with other species in wildlife monitoring (Birenbaum et al., 2022).

Facial recognition can be traced back to Woodrow Bledsoe whose research created a semi-automatic facial-recognition software where the facial coordinates were manually extracted from a photo (Bledsoe, 1964). These methods were improved upon and eventually succeeded by Eigenfaces which used principal component analysis (PCA). While the research on facial recognition continued into the 21st century, the technological advancements necessary to develop modern facial recognition systems did not emerge until 2011. The rapid improvement of facial recognition technology over the past 30 years was made possible not only through the proliferation of computing but also due to the availability of image datasets of increasing quality. With the invention of the graphics processing unit (GPU), deep learning has made great strides in computer vision and facial recognition. Deep learning improves machine learning by implementing multi-layered neural networks to extract hierarchical features from data. This is useful in the field of computer vision with the use of Convolutional Neural Networks (CNNs), which utilize supervised deep learning for image classification. The earliest model to employ a CNN architecture is considered to be LeNet which was developed to recognize handwritten digits. With the use of GPUs for deep learning, more complex architectures have been developed such as AlexNet, VGGNet, and others that have continuously pushed the envelope on facial recognition (Adjabi et al., 2020).

In recent years, Vision Transformers (ViTs) have emerged as a compelling alternative to traditional Convolutional Neural Networks (CNNs) in facial recognition tasks. ViTs utilize self-attention mechanisms to capture global relationships within an image, offering enhanced performance in various recognition challenges (Khan et al., 2022). Comparative studies have demonstrated that ViTs can outperform CNN-based models, such as YOLO and Faster R-CNN, particularly in scenarios requiring comprehensive context understanding (Dosovitskiy et al., 2021).

While human facial recognition has been a well-funded research topic for some time, it is no wonder that this area would be of great interest to humans considering its applications in security. Meanwhile, animal face recognition is less explored for obvious reasons - animals don't generally commit federal crimes or use technology that would necessitate biometric identification. Although animal recognition beyond binary classification can be of great use in wildlife studies, photo recognition could reduce the costs of tagging and tracking animals of interest for wildlife conservation (Birenbaum et al., 2022). Still, facial recognition for animals presents different challenges from human facial recognition. Recognition would be made difficult if features are occasionally obscured by the coat of fur with long-haired species. Additionally, attempting to train a single model across species could be problematic with the presence of large variations in facial structure present in animals that can occupy a single ecosystem. In a 2022 study, Birenbaum et al. repurposed PrimNet which was developed to identify primates to develop SealNet for the identification and tracking of harbor seals in Casco Bay, Maine. Animal recognition could also be extended to pets in the future for identification of lost pets or home security.

Another major development in deep learning for facial recognition is the rise of self-supervised learning, which allows models to learn robust feature representations from unlabeled data (Chen et al., 2021). This has proven useful in cases where obtaining labeled datasets is difficult, such as deepfake detection and forensic applications (Zhao et al., 2023).

The datasets used in this study were both sourced from Kaggle. Human faces were sourced from ‘Human Faces’ by A. Gupta who organized a collection of over seven thousand images of human faces and the ‘Celebrity Faces’ data set which were both collected via web scraping and augmentation of other datasets from Kaggle projects and competitions. Both males and females are included with photos generally capturing shoulders up in various lighting, angles, expressions, and ages. The dataset is described as having a thorough mix of races and age groups but with “special attention” to seniors. The dataset also has some computer-generated faces and the raw count of human faces before cleaning data set balancing was (n = 14,948).

The animal faces dataset, Animal Faces-HQ (Andrewmvd, 2020), was originally sourced from the StarGan v2 project (Choi et al., 2020) and consists of over 16,000 images split into three domains for cats, dogs, and wildlife. These photos generally similarly capture the animals to the human photos: shoulders up with varied lighting, angles, expressions, and ages. The ‘wildlife’ domain consists of various large cats, wolves, and foxes.

Recent applications of AI-based facial recognition in wildlife conservation have demonstrated its potential for identifying and tracking individual animals, aiding in anti-poaching efforts and behavioral monitoring (Parker, 2021). In controlled environments such as zoos and sanctuaries, automated monitoring systems utilizing AI-driven facial recognition help assess animal well-being and detect signs of illness or stress (Ultralytics, 2023).

This project aims to develop a deep-learning model capable of distinguishing between human and animal faces. To achieve this, a simple Convolutional Neural Network was trained on the labeled images extracted from the datasets of human and animal faces, treating this as a binary classification problem. This study explores CNN performance, challenges that were encountered during training, and the accuracy of the classification.

Methods

The Datasets

As mentioned previously, as part of preprocessing, the datasets the model was trained on were reduced in size from the original datasets which was done to ensure an equal distribution of human and animal faces. According to Wang et al., 2016, imbalanced data sets particularly when training neural networks can have severe negative impacts on model training as it leads to class bias. The images were systematically labeled and sorted for better training efficiency. The images in both datasets varied in pose, lighting conditions, and resolution. This variation introduced challenges in classification, as the model needed to generalize across different image qualities. Furthermore, images included various angles of faces, occlusions (such as sunglasses or animal fur), and variations in color and background. The three domains of the Animal Faces dataset were condensed to one, comprehensive, animal dataset. The images from the animal face dataset are at a resolution of 512 x 512. The human faces images are at varying resolutions (11 kB to over 2 MB) and file types (jpg, png, jpeg).

Preprocessing Steps

The renaming scheme was applied to systematically label images as either human_n.jpg or animal_n.jpg, ensuring the correct categorization and easy tracking of misclassified images. The split ratios were set to .7/.15/.15 for training/validation/testing. The number of total images

was limited to 14,630 due to the animal faces dataset having a smaller number of usable images. The images were randomly sampled from the datasets.

Pixel values were normalized to the range [0,1] to stabilize the training process by ensuring the input image data is on a consistent scale. Avoiding large inputs that occur on the typical RGB scale [0,255] can help prevent causing an unstable gradient by keeping the computations within a manageable range.

Data augmentation techniques were applied to prevent overfitting and improve the model's ability to generalize data. A shear transformation was used to shift part of the image along a specific axis, in this model, a 0.2 tilt (20%) was applied. Random zooming was implemented, adjusting images by up to 20%. The images were also randomly flipped horizontally to introduce even more variation (see Figure 1). All of these augmentations make the model more adaptable to small distortions and variations in the dataset, ideally improving performance. Also, the images were resized to 64x64 pixels to ensure consistency across the dataset.

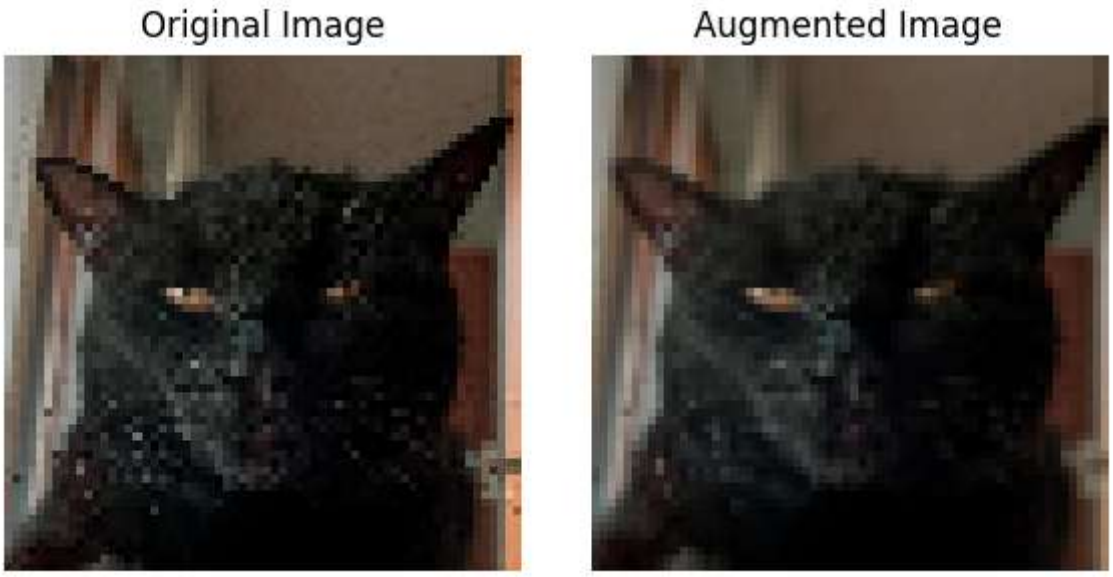
The datasets used to train the model were manually balanced by limiting the image count for each class (see Figure A1). This step was necessary to prevent bias in model predictions. The breakdown of image counts can be seen in Figure 2. A single prediction folder was created for inference testing. The purpose of this folder was to evaluate how well the model performed on unseen images outside of the training pipeline.

Advanced Data Processing

Post initial training and analysis, feedback was given by instructional staff. Specifically, it was suggested that along with our accuracy metric, we should add F1, precision, and recall

Figure 1

Demonstration of Image Augmentation



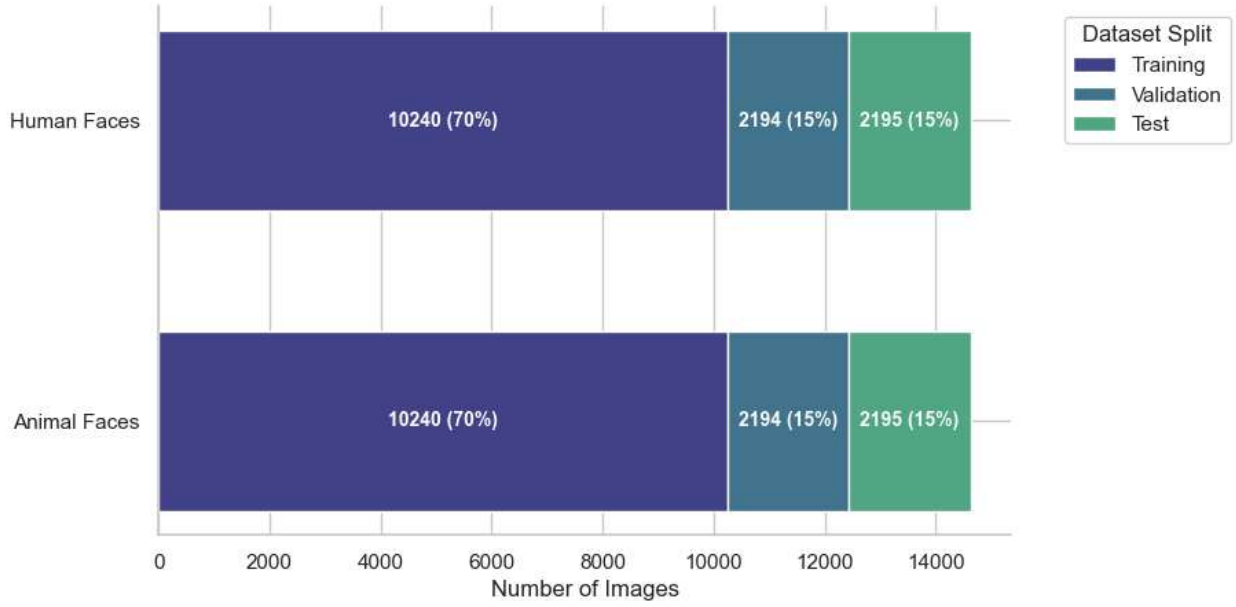
Note. Original image after resizing to 64 x 64 pixels (left) and image after random augmentation, shear transform and possibly zoom in (right).

metrics. It was also suggested that we add a confusion matrix. Accuracy is a commonly used metric in binary classification tasks, providing an overall measure of correct predictions relative to the total number of samples. Further, accuracy is defined as the ratio of correctly classified images to the total number of images processed by the model. However, accuracy has significant limitations, particularly when dealing with imbalanced datasets. In CNN-based binary image classification, a model could achieve high accuracy by predominantly predicting the majority class, leading to misleading results (Shahinfar et al., 2020). This is especially problematic in wildlife classification and medical imaging, where an imbalance in class distributions often occurs (Chicco & Jurman, 2020).

Concerning precision, it measures the proportion of correctly classified positive images

Figure 2

Dataset Split: Training, Validation, and Test for Animal vs. Human Faces



out of all images classified as positive. It is particularly useful in applications where false positives (FPs) carry a significant cost, such as in medical image analysis, where misclassifying a non-cancerous image as cancerous could lead to unnecessary interventions. In CNN-based image classification, precision ensures that only truly relevant images are classified as belonging to the positive class. However, it does not account for false negatives, which might be problematic in scenarios where missing a positive instance is critical (Powers, 2020). Recall, also known as sensitivity or the true positive rate, represents the proportion of actual positive images that were correctly classified. This metric is essential in scenarios where missing a positive instance is costly, such as detecting human faces in security applications or identifying anomalies in medical imaging. A high recall means that most positive instances are captured, though it can sometimes come at the expense of precision (Shahinfar et al., 2020).

The F1 score is the harmonic mean of precision and recall, balancing the trade-off between the two. It is particularly useful when both false positives and false negatives need to be minimized. In CNN-based binary classification, where dataset imbalance is a common challenge, the F1 score provides a more reliable measure than accuracy alone, as it considers both precision and recall rather than being skewed by class distribution. However, some researchers argue that the F1 score can still be misleading and advocate for alternative metrics such as the Matthews Correlation Coefficient (MCC), which considers all four elements of the confusion matrix (Chicco & Jurman, 2020).

Finally, in CNN-based binary classification, such as distinguishing between human and animal faces, the confusion matrix serves as the foundation for evaluating model performance by organizing predictions into true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). Accuracy, while commonly used to measure the proportion of correctly classified images $\left(\frac{TP + TN}{TP + TN + FP + FN}\right)$, can be misleading in imbalanced datasets where one class dominates, potentially inflating performance metrics (Shahinfar et al., 2020). Precision $\left(\frac{TP}{TP + FP}\right)$ helps mitigate this issue by measuring how many of the predicted human faces are correct, which is crucial in applications like biometric security (Powers, 2020). Conversely, recall $\left(\frac{TP}{TP + FN}\right)$ ensures that all human faces are detected, minimizing false negatives—an essential factor in surveillance and forensic applications (Shahinfar et al., 2020).

The F1 score $\left(\frac{2 \times (Precision \times Recall)}{(Precision + Recall)}\right)$ balances precision and recall, making it particularly useful in real-time facial recognition where both false positives and false negatives must be minimized (Chicco & Jurman, 2020). However, when dataset imbalance is present, the MCC offers a more reliable measure by accounting for all four confusion matrix components,

preventing misleading conclusions about model performance (Chicco & Jurman, 2020). Hence, the reason we ensured that all of our datasets were perfectly balanced before loading and training, is that this prevents bias towards the majority class and allows for a more accurate evaluation of model performance. By leveraging these metrics collectively, researchers can better analyze CNN performance, fine-tune models, and mitigate classification biases in human-animal face recognition tasks.

Thus, to enhance the model's performance evaluation, the additional metrics were implemented alongside accuracy. These metrics were introduced using TensorFlow's built-in Precision() and Recall() functions, while a custom F1-score function (F1Score) was created to provide a more comprehensive understanding of model performance during training. To ensure the model had enough time to learn intricate patterns, the training duration was extended from 25 epochs to 50. Despite this increase, early stopping was implemented to prevent overfitting by halting training when no improvements were observed in the validation F1-score for five consecutive epochs.

The highest validation F1-score (val_f1_m = 15.9589) was achieved in epoch 12, but subsequent epochs saw a decline in performance, with epoch 13 yielding a slightly lower F1-score of 15.7103. Due to early stopping with patience of 5, training continued for an additional five epochs after epoch 8, which marked the best validation F1-score. The model reverted to the weights from epoch 8, where the highest F1 score was recorded.

The initial confusion matrix revealed high numbers of False Positives (FP = 1118) and False Negatives (FN = 1096), indicating that the model was struggling to accurately classify both negative and positive samples. This pointed to potential issues with the thresholding mechanism. To address this, several preprocessing strategies were implemented, including changing the input

image size from 64x64 to 128x128 for better feature extraction, increasing the batch size to 64 for training efficiency, and introducing data augmentation techniques such as rotation, width, and height shifting, shearing, and zooming. These adjustments aimed to improve model robustness and reduce misclassifications.

Further experimentation focused on improving model stability and performance. This included implementing dynamic thresholding to replace the fixed 0.5 threshold and applying learning rate scheduling, which helped the optimizer converge more effectively. Additionally, L2 regularization (weight_decay=1e-5) was introduced to prevent overfitting by penalizing large weights. Despite these efforts, the second confusion matrix still showed significant misclassifications with high False Positives and False Negatives, indicating ongoing issues with the model's ability to distinguish between classes.

A thorough investigation into F1-score performance revealed persistent challenges, especially in achieving F1-scores above 0.50. Despite balanced data, which eliminated class imbalance as a factor, fine-tuning the model through various methods such as increasing batch sizes and experimenting with learning rates did not yield improvements. In the end, the decision was made to revert to the initial model setup, adding the precision, recall, and F1-score metrics as per the project requirements. Visual representations of the confusion matrix outputs during the training iterations can be found in Figure B1.

We also ensured that our preprocessing code was designed to optimize the model's performance across accuracy, F1 score, recall, and precision. For the validation and test sets, feature scaling was applied using ImageDataGenerator with a rescaling factor of 1./255, normalizing pixel values to the range [0, 1]. No augmentation was applied to the validation set to maintain the true distribution of the data. In contrast, the training set underwent both feature

scaling and data augmentation to reduce overfitting and improve generalization. Augmentation techniques included shearing, zooming, and horizontal flipping, which served to diversify the training data. The datasets were then organized into their respective sets, ensuring consistent image sizes (64x64) and batch sizes (32) for efficient training. This preprocessing approach aimed to enhance the model's ability to generalize, thereby improving its overall performance on various evaluation metrics.

To ensure the data was evenly distributed across classes, visual representations of the class distribution for the training, validation, and test sets were created. A helper function was implemented to generate count plots using Matplotlib and Seaborn, which displayed the number of images for each class (animal vs. human). The class labels were mapped to human-readable names, and the datasets were visualized using bar charts with annotated counts. This process confirmed that the data was balanced across the sets, helping to prevent issues with class imbalance that could negatively affect model performance. The visual representations of these distributions can be seen in Figure 2.

Advanced Mathematical Concepts

Before developing our final model, we experimented with the advanced mathematical concept of Grid Search, specifically Halving Grid Search, to find optimal hyperparameters for our model. This section discusses Halving Grid Search and how it was applied in our training experimentation. Halving Grid Search, a variant of the Successive Halving Algorithm (SHA), optimizes hyperparameter tuning by progressively eliminating underperforming configurations while allocating more resources to promising candidates (Jamieson & Talwalkar, 2016). The approach is particularly beneficial in deep learning image recognition, where training neural networks involves high-dimensional search spaces and expensive computational costs. By

systematically allocating minimal resources to a large set of hyperparameter configurations and then iteratively filtering out the worst-performing candidates, SHA accelerates model selection while maintaining high performance.

The method operates by dividing computational resources (e.g., number of training epochs, dataset size) among configurations, evaluating their performance, and discarding a fixed fraction of the least promising models at each iteration, thereby halving the search space. This structured elimination scheme efficiently narrows down the best hyperparameters for tasks like image classification in CNNs, reducing training overhead compared to an exhaustive grid search. Theoretical underpinnings suggest that SHA balances exploration and exploitation by progressively refining search quality without excessive resource waste, making it ideal for large-scale deep-learning applications (Li et al., 2020).

Concerning how we used Halving Grid Search in our training, first, we implemented Keras Tuner's Hyperband algorithm, an adaptive halving grid search method designed to optimize hyperparameter selection by allocating minimal resources to many configurations and progressively eliminating suboptimal candidates (Li et al., 2020). We initialized the tuner with "val_accuracy" as the objective function, set a maximum of 20 epochs, and applied a reduction factor of 3, meaning that only the top-performing one-third of configurations advanced to subsequent training rounds. To prevent overfitting and enhance efficiency, we incorporated an early stopping callback, which monitored validation loss and halted training if no improvement was observed for five consecutive epochs (Prechelt, 1998). We then executed the tuner.search() method using the training and validation datasets, allowing the model to systematically identify the optimal hyperparameters.

Next, we retrieved the best-performing hyperparameters by calling get_best_hyperparameters (num_trials=1). This process provided the most effective values for the number of filters in the first and second convolutional layers, the number of dense units in the fully connected layer, the dropout rate, and the learning rate. We printed these values to gain insight into the optimal configuration determined by Halving Grid Search. By refining the model with the most efficient hyperparameters, we ensured improved generalization, reduced training overhead, and computational efficiency (Bergstra & Bengio, 2012).

Since accuracy alone is often insufficient in evaluating model performance on imbalanced datasets, we then implemented a custom F1-score metric, which accounts for both precision and recall (Sokolova & Lapalme, 2009). We created the F1 Score class by extending tf.keras.metrics.Metric, allowing us to maintain internal variables for TP, FP, and FN. During each batch of training, these values were updated dynamically. As previously stated we computed the F1-score, we used the formula where precision and recall were derived from TP, FP, and FN values, and K.epsilon() which ensured numerical stability by preventing division by zero. This approach provided a more robust evaluation metric for binary classification tasks, particularly in cases where class imbalance could distort accuracy-based assessments (Flach, 2012).

Finally, we compiled the model using the Adam optimizer with a learning rate of 0.0005 and applied the binary cross-entropy loss function, which is standard for binary classification tasks (Kingma & Ba, 2015). We incorporated accuracy, precision, recall, and the custom F1-score as evaluation metrics, ensuring a comprehensive assessment of model performance and applied early stopping. The model trained on 20 epochs; however, with early stopping deployed, the training process was halted at epoch 16, and the model restored the weights from epoch 11,

which was identified as the optimal checkpoint. The final training loss was 0.0239, with a training accuracy of 0.9919.

The model demonstrated strong precision (0.9925), recall (0.9914), and an F1-score of 0.9919, indicating a well-balanced performance in classifying positive instances. On the validation set, the model achieved a validation accuracy of 0.9904, a validation loss of 0.0312, and an F1-score of 0.9905, demonstrating robust generalization. The precision-recall trade-off was evident, as the validation precision (0.9847) was slightly lower than the validation recall (0.9964), suggesting that while the model correctly identified most positive cases, it may have classified some negative cases as positives. The F1-score confirmed that there was a balance between precision and recall, with an optimal performance of 0.9905 on the validation set.

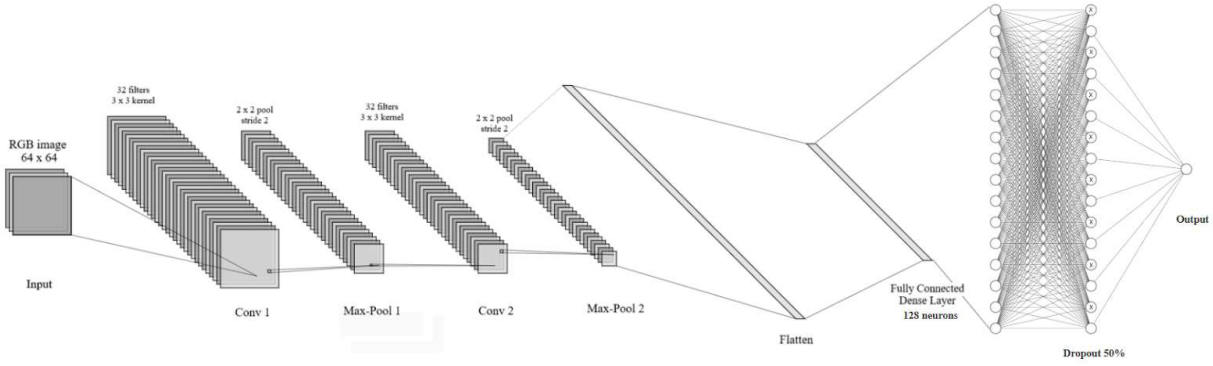
The Halving Grid Search method effectively optimized the hyperparameters, leading to high classification performance while mitigating overfitting through early stopping, thereby ensuring that the best weights were utilized for final evaluation. However, upon further analysis by our team, we concurred that while these outputs were optimal, they were not above or beyond what we accomplished with the version of our model that did not utilize the Halving Grid Search method. So, the decision was made by the team to move forward with our original model with some adjustments which are all discussed in the Model Architecture section below.

Model Architecture

The CNN model used in this study was implemented as a sequential architecture (see Figure 3), consisting of two convolutional layers, a flattening layer, and fully connected dense layers with dropout regularization. The architecture includes two Conv2D layers with 32 filters, a kernel size of 3, and ReLU activation. These layers extract important features from the input images before being passed to a Flatten layer. The fully connected layers consist of a Dense layer

Figure 3

CNN Architecture for Binary Image Classification



Note. This diagram illustrates the structure of the CNN architecture, depicting the multiple layers used for the classification of animal versus human faces. Diagram generated using *NN-SVG* (Lenail, n.d.) and modified to include additional neural network layers.

with 128 units and ReLU activation, followed by a Dropout layer with a rate of 0.5 to reduce overfitting, and a final Dense layer with a sigmoid activation function for binary classification.

This CNN architecture is fairly similar to LeNet-5 although the feature maps are different sizes. This framework has convolutional layers with 32 filters and size 3x3 while LeNet-5 has 6 kernels size 3x3. Layer three for LeNet-5 had 25 kernels that were 5x5 whereas this network repeats the filters from the first convolution. Here MaxPooling is used in favor of average pooling and there is only one fully connected layer with 50% dropout. This model also uses the Rectified Linear Unit (ReLU) for activation in the fully connected dense layer rather than the Tanh activation as in LeNet-5 (Lecun et al., 1998). ReLU is considered to be more appropriate for deep learning as it introduces non-linearity without being computationally expensive. The output layer utilizes a sigmoid activation for the binary output.

Training Configuration

The model training for this deep learning face recognition project was conducted using the Adam optimizer, selected for its adaptive learning rate capabilities. A batch size of 32 was used to balance memory efficiency and training speed, with the model trained for 50 epochs to ensure sufficient feature learning. To prevent overfitting and improve efficiency, an EarlyStopping callback was employed, monitoring validation loss with a patience of 5. Hyperparameter tuning was carried out through manual adjustments across different iterations to enhance model performance.

Results

Performance metrics were cataloged for both the validation and test sets in Table 1. Accuracy was recorded to provide a measure for correct classifications relative to the total amount of predictions. Precision and recall are included to gauge the percent of true over total positives and the percent of correctly predicted true positives. The F1 score was included to provide a measure that balances the precision and recall for a more holistic

Table 1

Performance Metrics from both Validation and Testing Sets

Metric	Validation Set	Test Set
Loss	0.1447	0.1447
Accuracy	0.9918	0.9918
Precision	0.9891	0.9891
Recall	0.9945	0.9945
F1	0.9918	0.9918

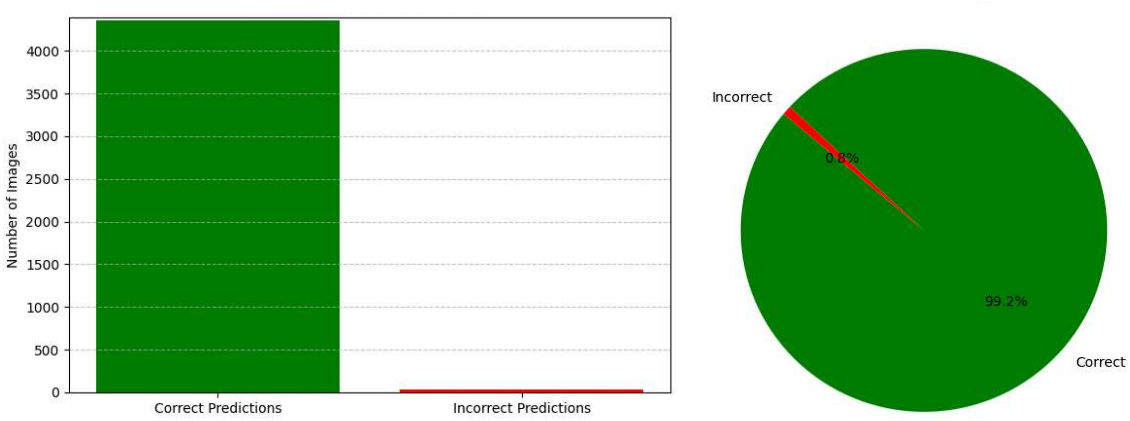
assessment.

The calculated metrics for both the validation and test sets are equal to the fourth decimal. These results are reasonable given the image sets were the same size which was not insubstantial at almost 2200 images per class. The performance of the model on the test set is displayed in Figure 4.

Training and validation accuracy (see Figure 5) follow a similar trend with rapid increase within the first five epochs and a gradual increase until the early stopping at 27 epochs. No major deviations to validation accuracy suggest there is no overfitting. For the loss curve, the model's training loss steadily decreased, showing effective learning. The validation loss initially decreased following the trend of training loss but had small fluctuations at the later epochs. No significant divergence was observed but the fluctuations may indicate overfitting.

Figure 4

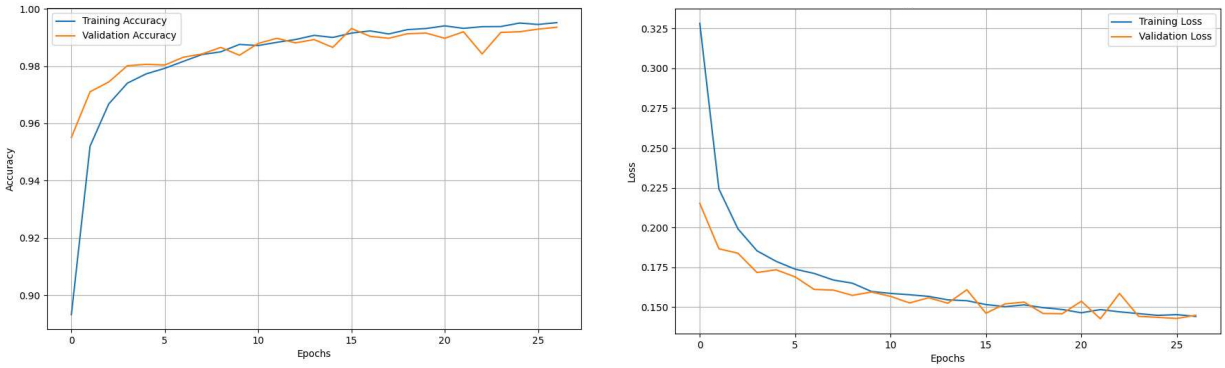
Performance of Original CNN



Note. The bar chart (left) illustrates the number of correct versus incorrect predictions made by the CNN. The pie chart (right) displays the percentage breakdown for the model's accuracy. The model achieved over 4000 correct predictions with an accuracy of 99.2%.

Figure 5

Model Training and Validation Curves

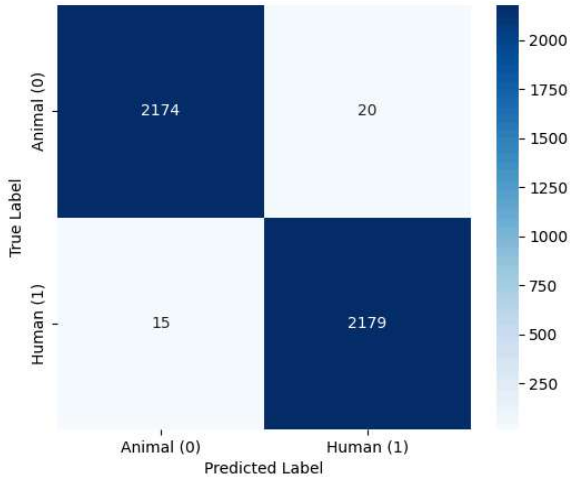


Note. Curves for model training and validation accuracy (left) and training and validation loss (right) over 27 epochs.

The confusion matrix (see Figure 6) indicates good model performance based on the validation set with only 35 total misclassifications. Twenty of those misclassifications were of animal images while human images only had 15 misclassifications.

Figure 6

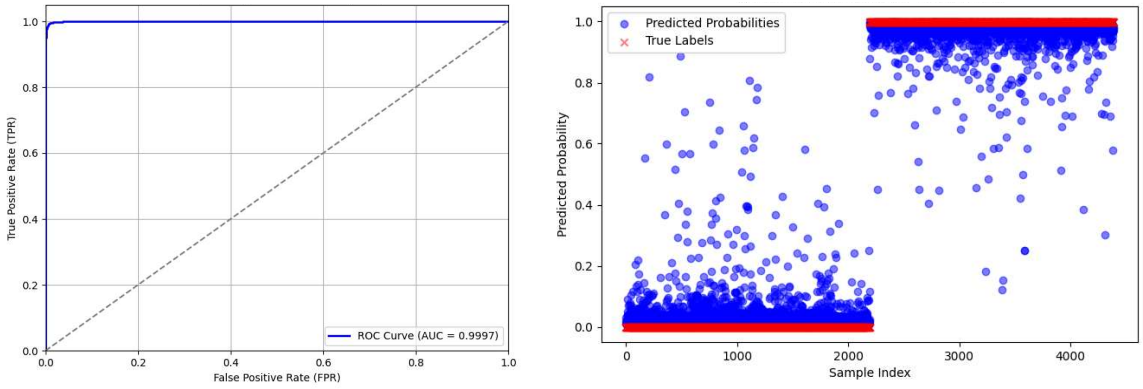
Confusion Matrix for Validation Set



The Receiver Operating Characteristic curve (ROC) (see Figure 7) was abnormal and suggested overconfidence of the model. This suspicion was confirmed with a scatter plot of predicted probabilities which showed a high concentration of predicted probabilities very close to 0 and 1 without many predictions splitting the difference. The single predictions made by the model with their referenced images show an accurate result (Figure 8). The model was correct on both accounts. Both images have good contrast and distinguishable features to the human eye. Figure 8 displays all images that were misclassified in the test set with 36 total images. The proportion of images remained similar with 24 misclassified animals and 12 misclassified humans. This indicates the model may have been overconfident in predicting the human label. Viewing the images, some possible causes leading to misclassification may have been age (as in the cases of the newborn humans or the puppy), dark or busy backgrounds, faces at angles greater than 45 degrees from the center, and poor contrast. The decrease in resolution also appeared to have a strange effect on some of the images.

Figure 7

ROC Curve and Scatter Plot of Predicted Probabilities



Note. The Receiver Operating Characteristic curve (ROC) for the validation set with AUC = 0.9997 (left) and a scatter plot of predicted probabilities (right) indicating overconfidence in the model.

Figure 8

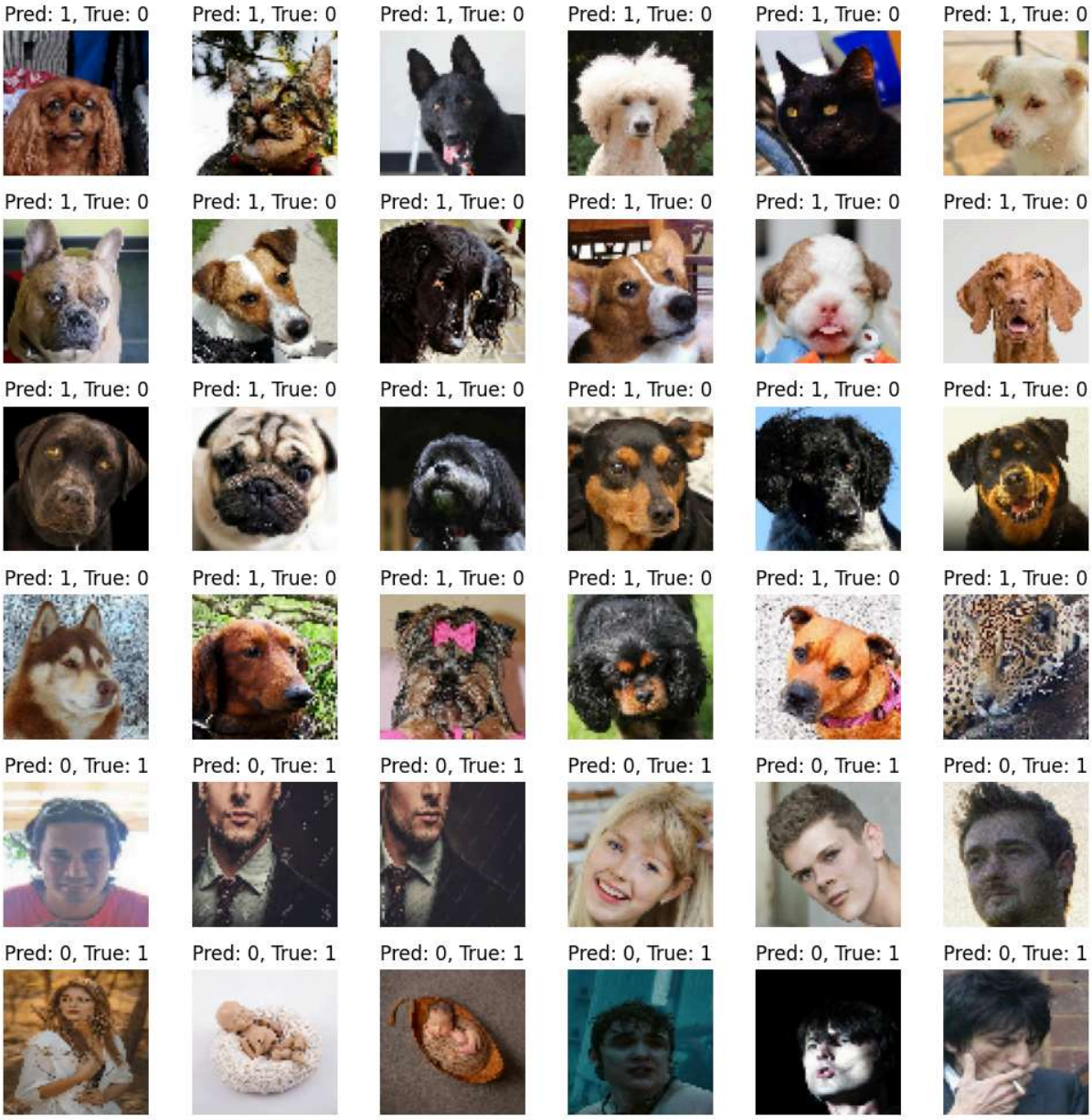
Predictions Made by the Mode on Select Images

Discussion

The CNN model employed was able to effectively distinguish human and non-human faces, achieving strong performance metrics. The model did not perform well when more severe distortions were introduced as well as with low contrast. These misclassifications also highlight challenges related to facial structure similarities, occlusions, and dataset bias. There was also an issue with overconfidence for both labels, but this issue was slightly more prevalent in human images. Introducing more augmentation, increasing the image resolution, and introducing more convolutional layers with different parameters could all help improve the model's performance. Steps were taken to mitigate the overconfidence that became apparent with the ROC curve, but the model did not perform well. The model's scalability to larger and more diverse datasets remains an open question, as performance may vary with increased class diversity.

Figure 8

Misclassified Images of the Test Set



Note. There are 24 animals (21 dogs, 2 cats, 1 large cat) and 12 humans (10 adults, 2 newborns).

Challenges in Model Development

There were a variety of challenges in the construction of this model. Variability in images is always a concern with computer vision as the different angles, occlusions, and lighting conditions make classification more complex. Overfitting was also an issue, this was addressed by applying dropout layers and introducing data augmentation.

Also of concern with any computer vision task are computational limitations. Training requires GPU acceleration due to dataset size and model complexity which can be slow if attempting to use cloud-based environments or less capable hardware.

An issue that occurred during post-processing was the production of an F1 value of around .5 for all epochs indicating random classification. This issue was compounded by the need to use a dated Tensorflow (v2.10.0 due to Windows) where the `F1Score` metric is not available. F1 calculation was initially regularizing TP/FP/FN across batches by calculating the mean (i.e. `self.tp.assign_add(tf.reduce_mean(tp))`). This was changed to simple addition which resolved the epoch level F1 calculation.

Initially, the confusion matrix was showing an F1 around 0.5 for all classes for the validation set as well. This was due to the `ImageDataGenerator.flow_from_directory()` function which automatically shuffles the data causing a mismatch of the classes. This was resolved by setting `shuffle = false` for the `flow_from_directory()` function for the validation set. A similar issue occurred with the test set and the same fix was applied.

An unresolved issue that occurred was with the ROC curve showing a single step rather than the desirable multi-step progression. This is likely due to overconfidence in the model as previously mentioned. The following steps were taken in attempting to fix this issue: label_smoothing increased from 0.05 to 0.1 to reduce overconfidence and the learning rate

reduced from 0.0005 to 0.0002 in an attempt to reduce the chance of overfitting. These changes led to a model that was not learning effectively (see Figure C1 and Figure C2). Loss was not decreasing properly and accuracy fluctuated but did not show a steady improvement throughout the epochs. This approach was abandoned.

Limitations and Bias

One limitation of the datasets had to do with the variety of animal species in the AFHQ dataset. This dataset only included cats, dogs, big cats (lions, tigers, cheetahs, etc.), foxes, and wolves. The model was not tested on any other animals so it may not perform well with the introduction of a novel species.

Another potential limitation is the makeup of the human faces dataset. Its documentation asserts that there is adequate diversity in the dataset but does not provide any metrics on sex, race, age, etc. As it can't be confirmed if the dataset is diverse, the model may not be generalizable for any subpopulation of humans.

Additionally, exhaustive measures were not taken in the search for optimal hyperparameters for the model due to the computational expense. The hyperparameters chosen performed well, but ideal parameters may have provided a better-tuned, less overconfident model.

Future Work

Some potential future directions for this project could include multi-class classification and real-world deployment. With a multi-class approach to classification, it would be possible to expand beyond binary classification to recognize specific animal species or subcategories of human faces could improve model robustness and generalizability. Future work could also

explore deploying the model in real-time applications, such as live video classification or integration into mobile applications.

Real World Applications and Impacts

Deep learning-based facial recognition has evolved beyond traditional security applications, finding use in various real-world scenarios that impact both human and animal identification. As AI-driven models continue to improve in accuracy, efficiency, and scalability, their deployment in sectors such as biometric authentication, wildlife conservation, home security, and real-time monitoring is expanding. However, integrating these technologies into practical applications comes with challenges, including computational constraints, data privacy concerns, ethical considerations, and the need for scalable models. Addressing these factors is crucial for ensuring reliable and responsible AI deployment. The following sections explore some of the key applications and their implications in different fields.

Feasibility of Real-Time Deployment

Deploying deep learning models for real-time applications necessitates addressing computational efficiency and latency. Edge computing has emerged as a pivotal solution, enabling data processing directly on devices rather than relying on centralized servers, thereby reducing latency and enhancing privacy (Wang et al., 2020). However, edge devices often have limited computational resources. To mitigate this, model optimization techniques such as pruning and quantization are employed to reduce model complexity without significantly compromising performance (Chen & Ran, 2019). Additionally, hardware advancements like NVIDIA's Jetson series offer affordable, high-performance solutions tailored for AI applications, making real-time, on-device inference more feasible (Wang et al., 2022).

Wildlife Monitoring and Conservation

Facial recognition technology is revolutionizing wildlife conservation by enabling non-invasive monitoring of individual animals. For species lacking unique markings, AI-driven facial recognition provides a reliable method for identification, facilitating studies on behavior and population dynamics (Clapham et al., 2020). Implementing these systems in remote areas presents challenges, including limited connectivity and power resources. Edge AI devices capable of on-site data processing address connectivity issues, while energy-efficient models ensure prolonged operation in the field (Wang & Liu, 2020).

Biometric Authentication and Security

Facial recognition has become integral to biometric authentication systems, from unlocking personal devices to securing sensitive facilities. Comparative analyses with advanced detection methods, such as YOLO and Vision Transformers (ViTs), are essential to ensure optimal performance in terms of speed and accuracy (Wang & Liu, 2020). Security applications demand rigorous evaluation of metrics like False Acceptance Rate (FAR) and False Rejection Rate (FRR), as misclassifications can have serious implications. Moreover, ethical considerations, including potential biases in AI models and privacy concerns, must be addressed to prevent unauthorized surveillance and data breaches (Clapham et al., 2020).

Pet Identification and Home Security

The application of facial recognition technology in pet identification is enhancing home security and pet care. Smart devices equipped with AI can recognize individual pets, allowing controlled access through pet doors and personalized feeding schedules. Ensuring accuracy across diverse breeds requires extensive and varied datasets. Privacy concerns are also paramount; implementing on-device processing safeguards user data by eliminating the need for cloud-based storage (Wang & Liu, 2020).

Scalability and Model Generalization

Scalability and generalization remain significant challenges in deploying facial recognition models across varied species. Expanding datasets to include a broader spectrum of animals enhances model robustness. Transfer learning techniques allow models to adapt to specific tasks, such as monitoring endangered species or assisting in veterinary diagnostics (Shukla et al., 2019). Real-world datasets often exhibit class imbalances. Adaptive loss functions, like focal loss or weighted cross-entropy, are employed to address these disparities, ensuring the model performs equitably across all classes (Wang & Liu, 2020).

Ethical Considerations

The deployment of AI-based facial recognition necessitates careful ethical deliberation. In human applications, biases related to race, gender, or age can lead to discriminatory practices, underscoring the need for diverse and representative datasets (Clapham et al., 2020). Privacy issues, particularly concerning unauthorized surveillance, pose significant risks. In wildlife conservation, while AI aids in monitoring endangered species, there is a risk of misuse, such as poachers exploiting tracking data. Implementing stringent data protection measures and access controls is essential to prevent such outcomes (Shukla et al., 2019).

Conclusion

This study successfully demonstrated that a Convolutional Neural Network (CNN) can distinguish between human and non-human faces with high accuracy. Despite challenges such as dataset variability and occasional misclassifications, the model performed well in binary classification. The findings suggest that CNN-based facial classification has promising applications in security, wildlife monitoring, and pet identification. Future work should focus on

improving dataset diversity and potentially exploring alternate model architectures to improve the generalizability of the model for real-world deployment.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., ... Zheng, X. (2015). *TensorFlow: Large-scale machine learning on heterogeneous systems*. https://www.tensorflow.org/

Adjabi, I., Ouahabi, A., Benzaoui, A., & Taleb-Ahmed, A. (2020). Past, present, and future of face recognition: A review. *Electronics*, *9*(8), 1188. https://doi.org/10.3390/electronics9081188

Andrewmvd. (2020). *Animal faces* [Data set]. Kaggle. https://www.kaggle.com/datasets/andrewmvd/animal-faces

Bledsoe, W. W. (1963, January). *A facial recognition project report*. Panoramic Research, Inc. https://archive.org/details/firstfacialrecognitionresearch

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research, 13*(1), 281–305. https://jmlr.csail.mit.edu/papers/volume13/bergstra12a/bergstra12a.pdf

Birenbaum, Z., Do, H., Horstmyer, L., Orff, H., Ingram, K., & Ay, A. (2022). SEALNET: Facial recognition software for ecological studies of harbor seals. *Ecology and Evolution*, *12*(5), e8851-n/a. https://doi.org/10.1002/ece3.8851

Chen, J., & Ran, X. (2019). Deep learning with edge computing: A review. *Proceedings of the IEEE, 107*(8), 1655–1674. https://doi.org/10.1109/JPROC.2019.2921977

Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2021). A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning*,

1597–1607. *arXiv preprint*, arXiv: 2002.05709. https://doi.org/10.48550/arXiv.2002.05709

Chicco, D., & Jurman, G. (2020). The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics, 21*, 6. https://doi.org/10.1186/s12864-019-6413-7

Choi, Y., Uh, Y., Yoo, J., & Ha, J.-W. (2020). StarGAN v2: Diverse image synthesis for multiple domains. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. https://www.kaggle.com/datasets/andrewmvd/animal-faces

Chollet, F., & others. (2015). Keras. GitHub. Retrieved from https://github.com/fchollet/keras

Clapham, M., Nevin, O. T., Ramsey, A. D., & Rosell, F. (2020). Automated facial recognition for wildlife that lack unique markings: A case study on brown bears. *Ecology and Evolution, 10*(7), 3504–3515. https://doi.org/10.1002/ece3.6840

Dagher, I., & Barbara, D. (2021). Facial age estimation using pre-trained CNN and transfer learning. *Multimedia Tools and Applications*, *80*(13), 20369-20380. https://doi.org/10.1007/s11042-021-10739-w

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2021). An image is worth 16×16 words: Transformers for image recognition at scale. *International Conference on Learning Representations*. *arXiv preprint*, arXiv: 2010.11929. https://doi.org/10.48550/arXiv.2010.11929

Gupta, A. (n.d.). *Human Faces* [Data set]. Kaggle. https://www.kaggle.com/datasets/ashwingupta3012/human-faces

Hallowell, N., Badger, S., McKay, F., Kerasidou, A., & Nellåker, C. (2023). Democratising or disrupting diagnosis? Ethical issues raised by the use of AI tools for rare disease

diagnosis. SSM. Qualitative Research in Health, 3, 100240–100240. https://doi.org/10.1016/j.ssmqr.2023.100240

Jamieson, K., & Talwalkar, A. (2016). Non-stochastic best arm identification and hyperparameter optimization. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS)* (pp. 240–248). PMLR. https://proceedings.mlr.press/v51/jamieson16.html

Khan, S., Naseer, M., Hayat, M., Zamir, S. W., Khan, F. S., & Shah, M. (2022). Transformers in vision: A survey. *ACM Computing Surveys, 54*(10s), 1-41. https://doi.org/10.1145/3505244

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition: Intelligent signal processing. *Proceedings of the IEEE*, *86*(11), 2278–2324. http://dx.doi.org/10.1109/5.726791

Lenail, A. (n.d.). *NN-SVG: Publication-ready neural network architecture schematics* [Computer software]. https://alexlenail.me/NN-SVG/LeNet.html

Li, L., Jamieson, K., Rostamizadeh, A., Gonina, E., Ben-Tzur, J., Hardt, M., Recht, B., & Talwalkar, A. (2020). A system for massively parallel hyperparameter tuning. *Proceedings of Machine Learning and Systems* (Vol. 2, pp. 230–246). https://proceedings.mlsys.org/paper_files/paper/2020/file/a06f20b349c6cf09a6b171c71b88bbfc-Paper.pdf

Parker, C. (2021). Facial recognition to save elephants from poachers. *Save The Elephants*. https://savetheelephants.org/news/facial-recognition-to-save-elephants-from-poachers/

Powers, D. M. W. (2020). Evaluation: From precision, recall and F-measure to ROC, informedness, markedness & correlation. *Journal of Machine Learning Technologies*,

2(1), 37–63. *arXiv preprint*, arXiv: 2010.16061. https://doi.org/10.48550/arXiv.2010.16061

Prechelt, L. (1998). Early stopping—but when? In G. B. Orr & K.-R. Müller (Eds.), *Neural networks: Tricks of the trade* (pp. 55–69). Springer. https://doi.org/10.1007/3-540-49430-8_3

Raschka, S., Liu, Y., Mirjalili, V., & Dzhulgakov, D. (2022). *Machine learning with PyTorch and Scikit-Learn: Develop machine learning and deep learning models with Python*. Packt Publishing.

Rescigno, M., Spezialetti, M., & Rossi, S. (2020, 12). Personalized models for facial emotion recognition through transfer learning. *Multimedia Tools and Applications*, *79*(47-48), 35811-35828. https://doi.org/10.1007/s11042-020-09405-4

Shahinfar, S., Meek, P., & Falzon, G. (2020). "How many images do I need?" Understanding how sample size per class affects deep learning model performance metrics for balanced designs in autonomous wildlife monitoring. *Ecological Informatics, 57*, 101088. https://doi.org/10.1016/j.ecoinf.2020.101088

Shukla, A., Cheema, G. S., Anand, S., Qureshi, Q., & Jhala, Y. (2019). Primate face identification in the wild. *arXiv preprint*, arXiv:1907.02642. https://doi.org/10.48550/arXiv.1907.02642

Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management, 45*(4), 427–437. https://doi.org/10.1016/j.ipm.2009.03.002

Ultralytics. (2023). Monitoring animal behavior using Ultralytics YOLOv8. *Ultralytics.* https://www.ultralytics.com/blog/monitoring-animal-behavior-using-ultralytics-yolov8

Wang, F., Zhang, M., Wang, X., Ma, X., & Liu, J. (2020). Deep learning for edge computing applications: A state-of-the-art survey. *IEEE Access, 8*, 58322–58336. https://doi.org/10.1109/ACCESS.2020.2982411

Wang, S., Liu, W., Wu, J., Cao, L., Meng, Q., & Kennedy, P. J. (2016). Training deep neural networks on imbalanced data sets. In *2016 International Joint Conference on Neural Networks (IJCNN)* (pp. 4368-4374). IEEE. https://doi.org/10.1109/IJCNN.2016.7727770

Wang, X., Khan, A., Wang, J., Gangopadhyay, A., Busart, C. E., & Freeman, J. (2022). An edge-cloud integrated framework for flexible and dynamic stream analytics. *arXiv preprint*, arXiv:2205.04622. https://arxiv.org/pdf/2205.04622

Zhao, H., Cui, H., & Wang, X. (2023). Exploring self-supervised vision transformers for deepfake detection. *arXiv preprint*, arXiv:2405.00355. https://arxiv.org/pdf/2405.00355

Appendix A

Data Splitting Code

Figure A1

Backend Code to Split Dataset into Train/Test Splits

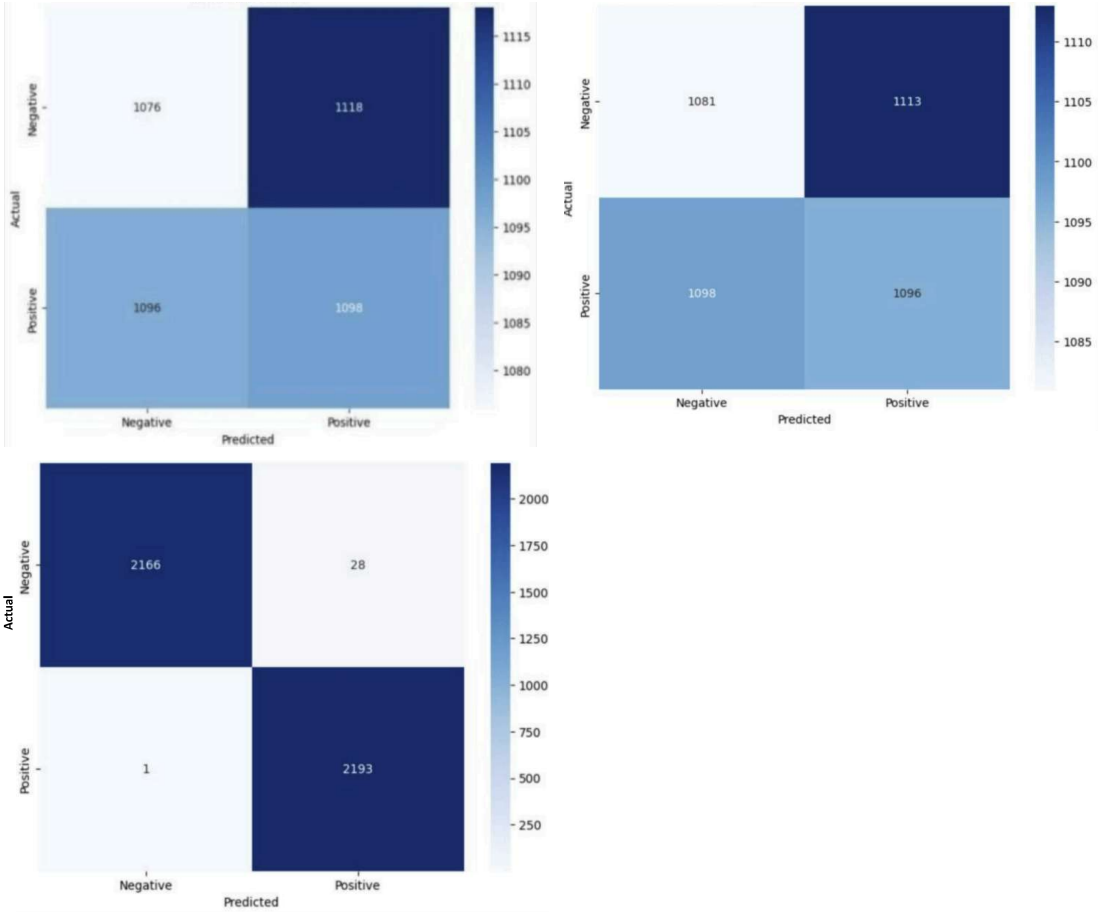
```python
import os
import shutil
import random

# Set correct paths based on your directory structure
base_dir = "/home/locutus/Downloads/CNN_Project_Group_17/Data"  # Folder containing all images
output_dir = "/home/locutus/Downloads/CNN_Project_Group_17/dataset"  # Output dataset folder

# Define classes and manually balance by taking the minimum count
min_images = 14630  # Equalizing to the smaller class (animal_faces)
classes = {
    "Animals": min_images,  # Matches "Animals" folder name
    "Humans": min_images    # Matches "Humans" folder name
}

# Define split ratios
train_ratio = 0.70
val_ratio = 0.15
test_ratio = 0.15

# Function to create necessary directories
def create_dirs():
    for split in ["train", "val", "test"]:
        for class_name in classes.keys():
            os.makedirs(os.path.join(output_dir, split, class_name), exist_ok=True)
    os.makedirs(os.path.join(output_dir, "single_prediction"), exist_ok=True)

# Function to split, rename, and move files
def split_data():
    for class_name, total_images in classes.items():
        source_folder = os.path.join(base_dir, class_name)  # Updated path for correct dataset structure
        images = os.listdir(source_folder)

        # Ensure we only use 'min_images' number of files for both classes
        images = random.sample(images, min_images)  # Select only min_images randomly

        # Rename the images with the prefix 'animal_' or 'human_'
        renamed_images = []
        for i, img in enumerate(images):
            extension = img.split('.')[-1]
            new_name = f"{class_name.lower()}_{i + 1}.{extension}"  # Renaming format
            os.rename(os.path.join(source_folder, img), os.path.join(source_folder, new_name))  # Rename image
            renamed_images.append(new_name)  # Keep track of renamed images

        # Pick one image for single prediction and remove it
        single_img = renamed_images.pop()  # Remove one image for the prediction set
        shutil.copy(os.path.join(source_folder, single_img), os.path.join(output_dir, "single_prediction", f"{class_name.lo

        # Compute split sizes after reserving single image
        train_count = int(train_ratio * (total_images - 1))
        val_count = int(val_ratio * (total_images - 1))
        test_count = int(test_ratio * (total_images - 1))

        # Assign images to respective splits
        train_files = renamed_images[:train_count]
        val_files = renamed_images[train_count:train_count + val_count]
        test_files = renamed_images[train_count + val_count:]

        # Move files to respective directories
        move_files(train_files, source_folder, os.path.join(output_dir, "train", class_name))
        move_files(val_files, source_folder, os.path.join(output_dir, "val", class_name))
        move_files(test_files, source_folder, os.path.join(output_dir, "test", class_name))

# Helper function to move files
def move_files(files, src_folder, dest_folder):
    for file in files:
        shutil.move(os.path.join(src_folder, file), os.path.join(dest_folder, file))

# Execute the data preparation
create_dirs()
split_data()

print("Data splitting complete. Class balancing applied. Folder structure created successfully!")
```

Appendix B

Confusion Matrices

Figure B1

Confusion Matrices Developed During the Secondary Model Development Process



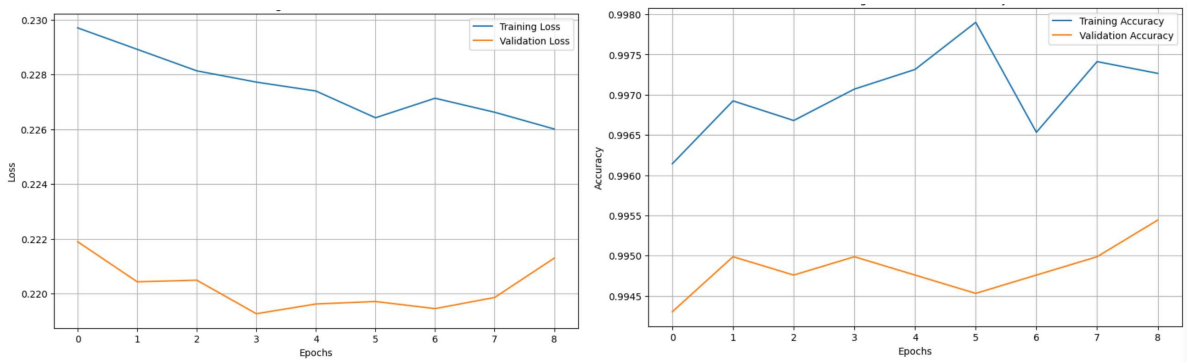
Note. These matrices were a product of updating the model with the feedback from the initial project submission. First, precision, recall, and F1-score metrics were implemented along with early stopping which was set with `patience = 5` (top left). Target and batch size were subsequently increased and changes were made to data augmentation (top right). More changes to tuning and batch size yielded the last matrix (bottom), but the notebook was abandoned due to a poor F1 score.

Appendix C

Impact of Adjustments on CNN Performance

Figure C1

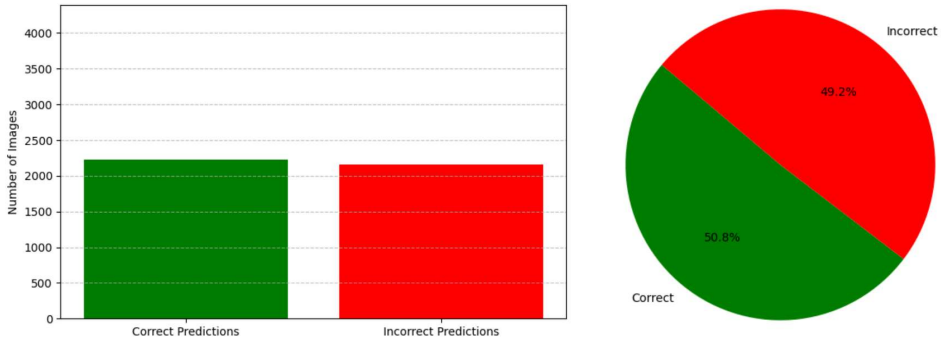
Impact of Increased Label Smoothing and Decreased Learning Rate on Accuracy



Note. Training and validation loss curves indicate that the model struggles to learn effectively (left). Training and validation accuracy show unstable performance with minimal improvement (right).

Figure C2

Impact of Increased Label Smoothing and Decreased Learning Rate on Predictions



Note. Model prediction performance shows nearly equal distribution between correct and incorrect predictions, highlighting ineffective learning (left). Model accuracy breakdown on the test set reveals a near-random classification (right).