Required Capstone Assignment 20.1 — Initial Report and EDA

Profile Photo Classifier: Human vs Avatar vs Animal

Report generated: September 01, 2025

# Executive Summary

We built a baseline image classifier to determine whether an Office 365 profile photo is a real human face, an avatar (stylized human face), or a non‑human (animal/other). The dataset was consolidated from FairFace, a Kaggle rehost of Google’s Cartoon Set, and Kaggle Dogs vs Cats. After cleaning, deduplication, and stratified splits, a frozen ResNet50 backbone with a lightweight head achieved strong validation/test results (see classification report and confusion matrix). Common errors: highly photorealistic avatars mistaken for humans and distant humans in cluttered scenes.

Notebooks in repo root: [LoadDataset.ipynb](https://github.com/lobral2728/ucb_ml_capstone/blob/resnet/LoadDataset.ipynb) (data & EDA) and [UCB\_ML\_Capstone.ipynb](https://github.com/lobral2728/ucb_ml_capstone/blob/resnet/UCB_ML_Capstone.ipynb) (modeling & evaluation).

# Business Context

My company is encouraging people to add their pictures into their Office365 profiles. Since the company is large and geographically spread out, the primary mode of communication is Teams chats. Having a picture in your profile is a way to increase the familiarity and connection between people. Currently, less than 50% have their picture up. There are thousands of pictures of pets, cartoon faces, cars, landscapes, and many other creative expressions but not helpful in achieving the goal. Ultimately using a model to determine if a person has a human picture or a picture of something will create a cost-effective way to track if we are closing in on the goal.

Data Sources

Ultimately, the Microsoft Graph API will be used to mine profile pictures to classify them as human or not. For the purpose of training the model, I will be using the FairFace Links to an external site.dataset. This dataset has a balanced set of faces in terms of race, age, and gender. I plan to add a dataset from Kaggle that contains animal faces Links to an external site.to complete the dataset. The training set will be approximately 10,000 human faces and 10,000 animal faces. I may also utilize other picture types depending on the results.

Techniques

I plan to heavily rely on Convolutional Neural Network (CNN) modeling utilizing the ResNet50 pretrained model as foundation. In my trails thus far, building my own CNN from scratch is proving very difficult. Fine-tuning an existing CNN is the preferred method in this case.

Expected Results

By utilizing a pretrained model and fine-tuning it with the dataset described above, I expect to be able to build a binary classifier that can detect when a picture is a human face. I also expect the results to be consistent across race, gender, and age.

Why This is Important

Currently, the reporting on this initiative is taking a person one day per reporting period to create this report by hand. With the model in place, that time can be significantly reduced. The job for a person then becomes only evaluating only the pictures where the model was not highly certain. Additionally, the ultimate goal of getting people more familiar with their peers can pay off in less quantifiable ways.

# Project Organization

• This Word document summarizes findings and references the two Jupyter notebooks.

• Notebooks contain headings and explanatory markdown for each major step.

• No unnecessary large binaries are tracked; raw data are excluded via .gitignore.

• Final splits live under `data/final/{train,val,test}/{human,avatar,animal}/` with sensible names.

# Syntax and Code Quality

• Correct imports/aliases; helper functions include comments and clear variable names.

• Code executes without errors after recent fixes; long framework logs are suppressed.

• Demonstrates competency with pandas and visualization (Matplotlib).

• TensorFlow/Keras used idiomatically for input pipelines and modeling.

# Visualizations

• Class distribution plots per split with readable labels and titles.

• Sample image grids showing post‑resize/pad results to verify no cropping issues.

• Aspect‑ratio and sharpness histograms for outlier inspection.

• Training curves (loss/accuracy), confusion matrix, and a 24‑image test gallery with labels, predictions, and probabilities.

# Data Cleaning and EDA

• Removed corrupt/unreadable files and standardized channels to RGB.

• Deduplicated with SHA‑1; screened near‑duplicates with perceptual hash (pHash) to prevent leakage across splits.

• Feature engineering: unified labels {human, avatar, animal}, normalization, augmentation (flip/rotate/color jitter).

• Outliers: extreme aspect ratios diagnosed; switched to `tf.image.resize\_with\_pad` to preserve content.

• Stratified splitting with a fixed seed; class weights computed from the training distribution.

# Modeling (Baseline)

• Model: ResNet50 (ImageNet, include\_top=False) \*\*frozen\*\*; head = GlobalAveragePooling → Dropout → Dense(3, softmax).

• Rationale: Transfer learning provides robust features; a frozen backbone gives a clean baseline without fine‑tuning.

• Optimization: Adam (1e‑4), SparseCategoricalCrossentropy; callbacks = EarlyStopping, ReduceLROnPlateau, and save‑best‑only.

• Metrics: Accuracy plus macro‑averaged F1 and per‑class precision/recall/F1 (classification report) on a stratified test set.

• Interpretation: Confusion matrix and gallery reveal failure modes (photorealistic avatars; small/distant humans).

# Key References

- ResNet: He et al., 2016 — https://arxiv.org/abs/1512.03385

- Keras Applications (ResNet50) — https://keras.io/api/applications/resnet/

- Keras Transfer Learning Guide — https://keras.io/guides/transfer\_learning/

- TensorFlow: Mixed Precision — https://www.tensorflow.org/guide/mixed\_precision

- Keras Preprocessing/Augmentation Layers — https://keras.io/api/layers/preprocessing\_layers/

- Dropout: Srivastava et al., 2014 — https://jmlr.org/papers/v15/srivastava14a.html

- Class Weights (scikit‑learn) — https://scikit-learn.org/stable/modules/generated/sklearn.utils.class\_weight.compute\_class\_weight.html

- Data Leakage via Duplicates (Barz & Denzler, 2020) — https://arxiv.org/abs/1902.00423