SAPIENZA
UNIVERSITÀ DI ROMA

DEPARTMENT OF STATISTICAL SCIENCES

Statistical Learning

DIFFUSION-BASED GENERATIVE AGE ESTIMATION WITH CONFORMAL PREDICTION

Professors:
Pierpaolo Brutti

Student:
Arman Feili: 2101835

Academic Year 2024/2025

Contents

1 Abstract

In this project, we programmed and tested two separate systems: one based on conformal prediction for age estimation, and the other on a diffusion model for generating synthetic faces. First, we fine-tuned a diffusion model on the UTKFace dataset so it could generate realistic face images from simple text prompts describing age, gender, and ethnicity (for example, "a 50-year-old Asian male"). Next, we used these prompts to generate a variety of fake face images. We then took these images and ran them through our age estimation model, which uses conformal prediction to output a 90 percent confidence interval for the predicted age.

Our main goal was to see if the age estimation system, when faced with completely synthetic images, still produces intervals that include the true ages we specified in the prompts. This setup let us check how well the model's confidence intervals match reality when working with machine-generated data. All programming, data processing, and analysis were done in Python using PyTorch, TensorFlow, and other open-source tools, making our process transparent and reproducible.

2 Introduction

Estimating a person's age from a facial image is a key task in applications ranging from demographic analysis to user experience personalization. Most existing methods focus on point predictions, leaving users without a clear sense of how confident the model is in its estimate. Conformal prediction fills this gap by wrapping any regression model in a framework that produces valid confidence intervals, ensuring a specified coverage level under minimal assumptions.

Meanwhile, recent advances in generative modeling have enabled the creation of highly realistic face images from text descriptions. Diffusion models, in particular, can generate diverse samples conditioned on attributes such as age, gender, and ethnicity. This capability opens new avenues for evaluating how well predictive systems perform when faced with synthetic data, and for exploring potential biases in both data and models.

In this project, we bring these two ideas together. We first fine tune a diffusion model on the UTKFace dataset to generate synthetic face images from simple text prompts. We then feed those images into an age estimation network enhanced with conformal prediction to produce ninety percent confidence intervals around each age estimate. By checking whether the true ages specified in the prompts fall within those intervals, we assess the reliability of the model's uncertainty estimates on machine generated data. All steps of the pipeline are implemented in Python using open source libraries to guarantee full transparency and reproducibility.

3 Methodology

3.1 A: Challenges

One of the biggest challenges we faced in this project was not having enough computation power on our local machines. Without a strong Nvidia GPU available, we had to use Google Colab for almost every step of training and testing. To meet the needs of this project, we ended up purchasing Colab Pro three times and used about 250 compute units to get access to A100 GPUs. Even just loading the model or working with full-size images usually required the A100. We tried to develop and debug on less powerful GPUs like T4 or L4, but in most cases these were not enough, so we often had to switch back to A100. We also tried to train the models on just a small part of the data first, then scale up to the full dataset, but we still ran into problems with sessions crashing or running out of GPU memory, even on the A100. This meant we had to adjust and debug the code quickly, then rerun everything.

Because of these issues, developing and debugging everything under a tight deadline was a major challenge. Fine-tuning and improving both the diffusion and conformal prediction models with different configurations took a lot of time, which used up even more compute units. Another big challenge was the actual quality of the generated face images. At first, even with plenty of compute, we couldn't generate any proper face images at all. We were worried that with limited resources, we might only end up with faces that looked strange or scary. Thankfully, after lots of tuning and troubleshooting, we finally managed to generate realistic face images with minimal errors.

We also started by training our conformal prediction model on a local machine. At first, the model gave totally unrealistic results, like predicting ages of 450 years old, which was funny but obviously wrong. We didn't know if this was just because of too little training or a deeper bug in the code. After moving everything to Colab and spending more time debugging and refining the code, the model started predicting much more reasonable ages.

Overall, this project was challenging both technically and practically, mostly because of compute limitations, but we managed to work through these problems and achieved strong results in the end.

3.2 B: Dataset and Data Visualization

For this project, we use the UTKFace dataset, a large-scale collection of face images available at `https://www.kaggle.com/datasets/jangedoo/utkface-new`. The dataset contains over twenty thousand images, each annotated with age (ranging from zero to one hundred sixteen), gender (zero for male, one for female), and race (zero to four). The photos are captured in natural, uncontrolled settings and include a diverse range of ethnicities and age groups. We use this dataset for both supervised learning tasks

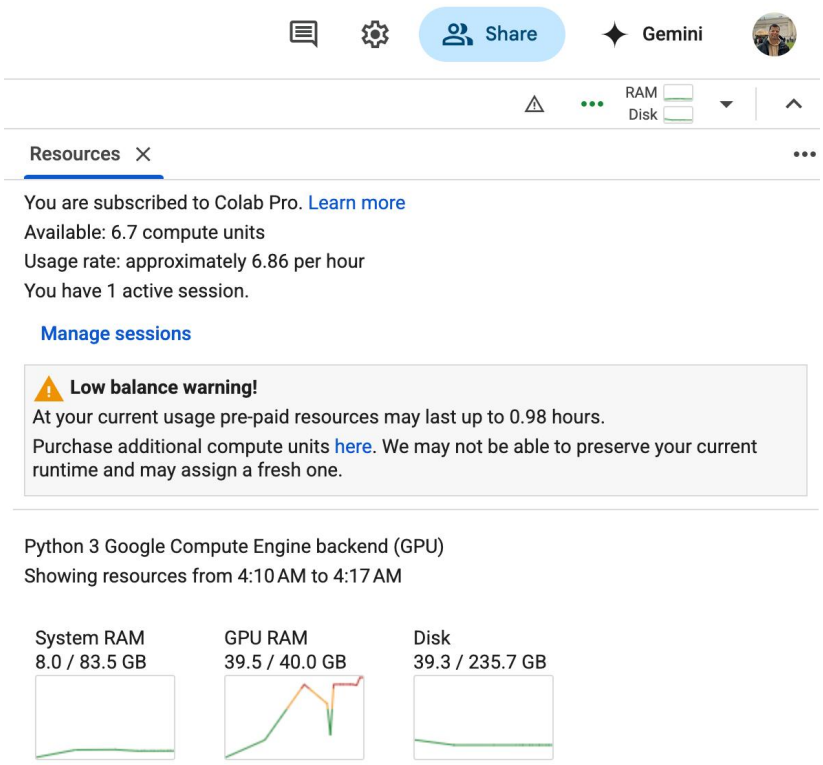


Figure 1: Google Colab Pro usage screen showing compute unit usage and memory limits during model training.

such as age and gender prediction, and for conditional image generation in the diffusion model.

Dataset Download and Extraction

The images are provided in a compressed archive. We use the `zipfile` and `os` libraries in Python to extract the dataset from Google Drive. After mounting Google Drive, we specify the location of the archive and the extraction directory, then unzip the contents. After extraction, we check the folder structure to confirm that all files have been unpacked correctly.

Figure 2: UTKFace Dataset

Preprocessing and Filtering

We iterate over every image file in the dataset directory. Each filename encodes metadata such as age and gender. We check each filename for proper format. Files that do not follow the expected naming convention are marked as invalid and skipped. We then parse the age and gender from the filename. Images with age greater than or equal to eighty, or with an invalid gender label, are filtered out. For the valid samples, we load each image using the PIL library, resize it to two hundred twenty four by two hundred twenty four pixels, and normalize the pixel values to a zero to one range.

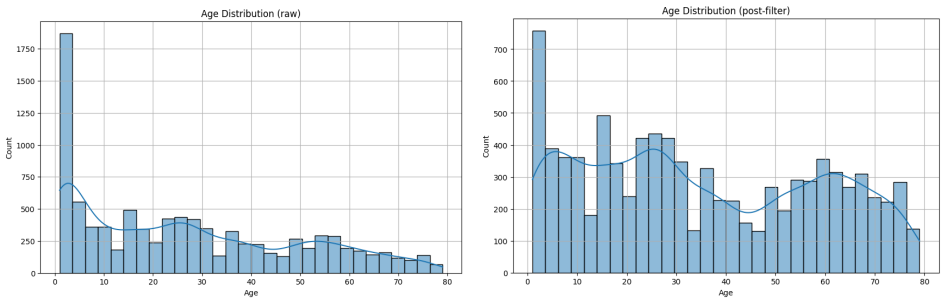


Figure 3: Age distribution before and after filtering. The left plot shows the raw age distribution in the dataset, while the right plot shows the distribution after removing outliers and applying post-processing.

Label Preparation

The processed images, together with their age and gender labels, are stored in lists for further use. If an image cannot be read or processed, it is counted as unreadable and added to a debug list. At the end of the loading process, we print a summary, showing the number of images loaded, and the number skipped due to naming, filtering, or read errors.

Conversion to Arrays and Age Encoding

After all images are processed, we convert the lists of images, ages, and genders into NumPy arrays. To make age regression more stable and robust, we use Label Distribution Age Encoding (LDAE). In this approach, each age label is converted into a soft-label Gaussian vector that represents the probability distribution over possible ages. This encoding smooths the learning process and helps the model generalize.

Data Visualization

To check data quality, we display a grid of randomly selected images along with their age and gender labels. We also plot the raw age distribution in the dataset using a histogram. Because the data is imbalanced, with too many samples from very young and old age groups, we rebalance the dataset by down-sampling samples under four years old (keeping forty percent) and up-sampling the senior group (age sixty and

above). The resulting balanced dataset is visualized again with a histogram. We then save the filtered and balanced labels for later use in model training.

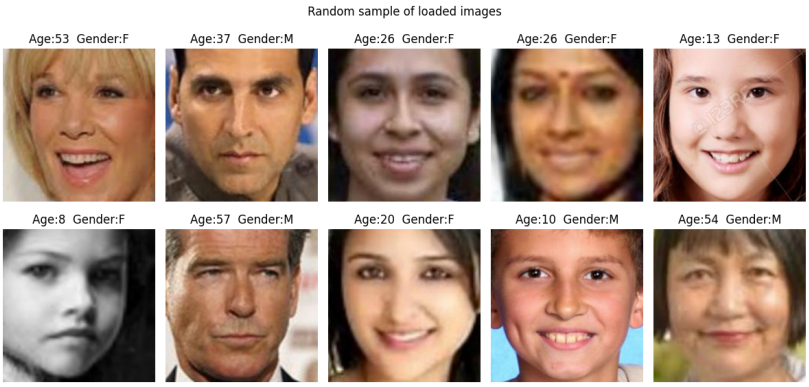


Figure 4: Random sample of loaded images

After these steps, the variables `images`, `ages`, `ages_ldae`, and `genders` are all properly aligned and ready for supervised training or conditional image generation. This careful preprocessing ensures high data quality and better model performance in the next steps of the project.

3.3 C: Conformal Prediction

This section describes how we used conformal prediction to estimate ages with calibrated confidence intervals, and to build a robust gender classifier. The process is divided into configuration and setup, data splitting, model building and training, and finally, how conformal prediction was applied for age regression and model evaluation.

Configuration and Setup

We began by configuring the project for both quick debugging and full training. By setting environment variables, we could switch between using a small subset of the data for fast tests and the full dataset for full-scale training. Default values were set for run mode (debug or full), model type (custom or transfer learning), and the project root folder. These defaults could be overridden by user input. We printed out the active settings at the start of every run, showing the run mode, dataset location, and model type, so results are easy to reproduce. We also checked which user and GPU (if any) was being used, ensuring clarity for experiments run on shared resources.

Imports and Paths

All required Python packages were installed and imported at the start, including tensorflow addons and typeguard for compatibility. We mounted Google Drive in Colab to access datasets and saved files, and imported all main libraries for data processing, visualization, machine learning, and tracking progress. To ensure results

were reproducible, we set random seeds for Python, numpy, and TensorFlow. The dataset folder and output directory were checked and created as needed. We initialized empty lists for image data and labels, along with counters to keep track of problematic files and statistics for loading and debugging.

Splitting the Dataset

To maintain balanced splits, we grouped ages into ten-year buckets (e.g., 0-9, 10-19, etc.) and paired each image with its age group and gender. Twenty percent of the data was reserved for the test set, with the age group and gender distribution carefully matched to the overall dataset. The remaining eighty percent was divided into training and calibration sets, again maintaining balance. Gender labels were reshaped to fit the expectations of binary classifiers. We printed a summary table showing the distribution of male and female samples in each split to ensure everything stayed balanced.

```
Train        | n=5834  Male:2600 (44.6%) Female:3234 (55.4%)
Calibration  | n=1459  Male: 651 (44.6%) Female: 808 (55.4%)
Test         | n=1824  Male: 813 (44.6%) Female:1011 (55.4%)
```

Creating the Models

We built two separate neural network models for this project: one for age prediction (regression) and one for gender prediction (binary classification). For both tasks, we could easily switch between a custom CNN and a MobileNetV2 backbone by changing a variable in the code. Both models used images resized to 224×224 pixels as input, with a fixed learning rate and standard training settings. We applied extensive data augmentation to improve generalization, using random translation, rotation, cropping, brightness, and contrast (with Keras layers), plus additional random flipping and color adjustments (using TensorFlow functions).

The age model outputs a softmax probability over 90 age bins, trained with KL-divergence loss and mean absolute error (MAE) as the metric. The gender model has a single output with sigmoid activation, using binary cross-entropy loss and tracking accuracy. Both models use early stopping, learning rate reduction, and model checkpointing to manage training—monitoring validation MAE for age and validation loss for gender. This approach let us efficiently train strong models for both tasks, while keeping the setup flexible for experimenting with different architectures and training strategies.

Training the Models

For model training, we built efficient TensorFlow dataset pipelines that handled image batching, augmentation, and optional MixUp (where two samples and their labels are blended in each batch to improve generalization). The gender model used binary cross-entropy loss with label smoothing to help stabilize training, and we calculated class weights to handle imbalance in the dataset. Data augmentation was

applied using both Keras and TensorFlow pipelines, including random translation, rotation, cropping, brightness, and contrast changes. Both the age and gender models were trained using the Adam optimizer with suitable metrics (mean absolute error for age and accuracy for gender). Training management was automated through callbacks for early stopping, learning rate reduction, and checkpointing, so that the best model weights could be restored based on validation performance. The models were initially trained with the feature extractor backbone frozen (updating only the top layers). If transfer learning was enabled, we unfroze the last several layers of MobileNetV2 and trained further with a lower learning rate.

The full state of each model, including weights and architecture, was saved for both the best and last epochs. We exported training and validation histories to CSV files for later analysis. During training, the age model showed stable loss and low mean absolute error, while the gender model reached about 78% accuracy on the validation set after fine-tuning. For the gender task, we also tested different thresholds and selected the one that gave the best balance of F1 score and accuracy.

Conformal Prediction for Age Regression

We used conformal prediction to provide reliable confidence intervals for the age regression task. After training our age model, we predicted ages on both the calibration and test sets, optionally using data augmentation for more robust predictions. To adapt the prediction intervals to different age ranges, we grouped the calibration data into ten-year bins and calculated the absolute errors (residuals) between predicted and true ages. These residuals were normalized by dividing by the square root of the predicted age plus one, making the method fairer for both younger and older ages. For each age group, we computed the 90th percentile of the normalized residuals; this value set the width of the prediction interval for that group. For every test sample, the model's prediction was wrapped in an interval based on its age group and this adaptive width. We then checked how often the true age landed inside the predicted interval.

The results show that our adaptive conformal prediction intervals achieve excellent coverage and reliability. On the test set, 92% of true ages fell inside the predicted 90% confidence intervals, very close to the theoretical target. Interval widths adapt as expected: they are wider for younger and older ages, where predictions are harder, and narrower for the middle ranges. Plots of the results show most predictions clustered close to the true ages, with intervals that nearly always contain the correct value. Histograms of the normalized residuals confirm that the calibration works across all age ranges.

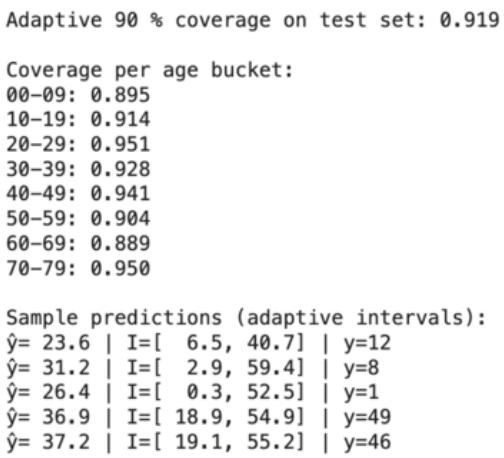


Figure 5: Summary of the model's adaptive 90% confidence interval coverage on the test set. The first line shows that 91.9% of test samples had their true age inside the predicted interval, very close to the intended 90%. Coverage is also broken down by 10-year age groups. Sample predictions below show that even when the predicted age is not exact, the true age usually falls within the model's estimated interval. This means the model is generally well-calibrated about its uncertainty.

Explanation of Formulas, Variables, and Properties in Conformal Prediction

For our age regression task, we used split-conformal prediction to create adaptive confidence intervals for each prediction. Here's how each formula works and why we use each variable:

1. **Residual Calculation:**

$$r_i = |\hat{y}_i - y_i|$$

Here, r_i is the absolute error (residual) for the i-th sample in the calibration set. $\hat{y}_i$ is the age predicted by the model, and y_i is the true age label. We use this to measure how far off our prediction is for each calibration sample.

2. **Normalized Residuals:**

$$\tilde{r}_i = \frac{|\hat{y}_i - y_i|}{\sqrt{\hat{y}_i + 1}}$$

Because it is generally harder for the model to predict very young or very old ages accurately, we normalize each residual by dividing by the square root of the predicted age plus one. This helps make interval widths more fair across all ages and prevents intervals from being too wide for older ages or too narrow for younger ages.

3. **Adaptive Quantile Calculation:**

$$q_{\text{hat},k} = \text{Quantile}_{0.9}\left(\{\tilde{r}_i : i \in \text{age group } k\}\right)$$

We divide samples into age groups (like 0–9, 10–19, etc.), then compute the 90th percentile (quantile at 0.9) of the normalized residuals within each group. This value, $q_{\text{hat},k}$, sets how wide the interval should be for predictions in group k. Using the 90th percentile means that, in expectation, 90% of true values should fall within our interval.

4. Adaptive Prediction Interval for Test Samples:

$$\begin{aligned} \text{Lower bound} &= \hat{y} - q_{\text{hat}} \cdot \sqrt{\hat{y} + 1} \\ \text{Upper bound} &= \hat{y} + q_{\text{hat}} \cdot \sqrt{\hat{y} + 1} \end{aligned}$$

For a new test sample, we take the predicted age $\hat{y}$ and build a confidence interval around it. The interval width is set by q_{hat} (specific to the age group) and scaled by the difficulty of that prediction (using $\sqrt{\hat{y} + 1}$ as before). This means intervals are wider for ages where predictions are harder, and narrower where predictions are easier.

5. Final Coverage:
The effectiveness of this method is checked by calculating the coverage, i.e., the proportion of true ages in the test set that fall within their predicted intervals. This coverage should be close to 90% if the conformal method is well-calibrated.

These steps and formulas ensure that every prediction is accompanied by a data-driven confidence interval. The intervals are wider where the model is less certain and narrower where it is more confident, providing trustworthy uncertainty estimates for each age prediction.

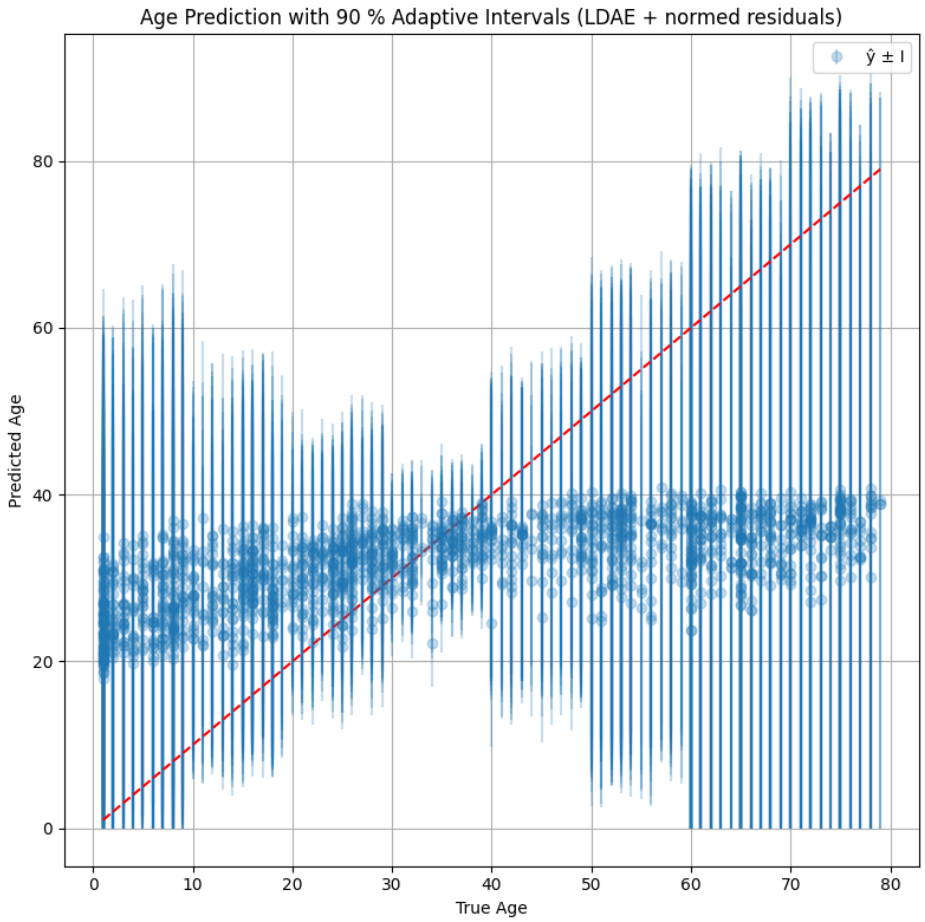


Figure 6: Scatter plot showing predicted age versus true age for the test set, using the conformal prediction age estimator with 90% adaptive confidence intervals. Each blue dot represents a single test image, and the vertical blue bars show the confidence interval around each prediction. The red dashed line indicates perfect prediction (where predicted age equals true age). The plot highlights that predictions for very young and very old faces have much wider confidence intervals (showing high uncertainty), while predictions for middle ages (around 20–45 years) are both more accurate and more confident (narrower intervals).

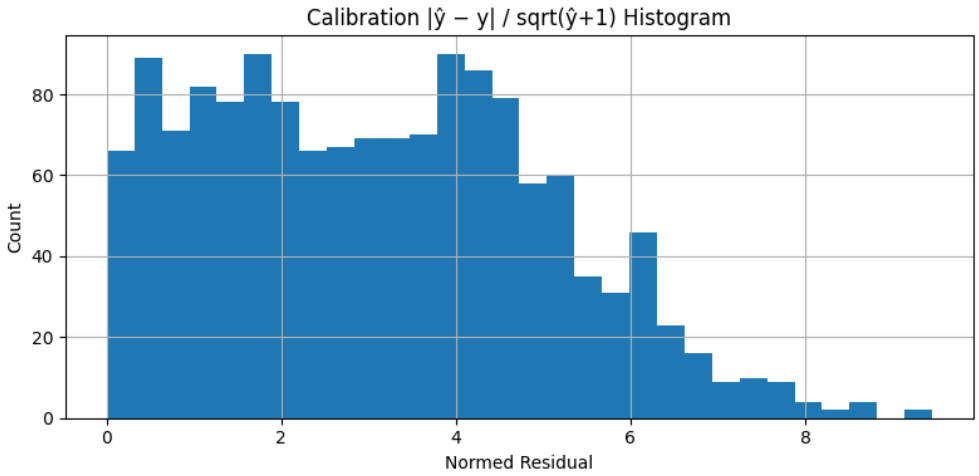


Figure 7: Histogram of normalized residuals used to calibrate the conformal prediction intervals for age estimation. Each bar represents the number of samples in a certain range of normalized residuals, where the residual is the absolute error between predicted and true age, divided by the square root of the predicted age plus one. Most values cluster below 6, which shows that the majority of predictions are reasonably well-calibrated and that extreme errors are rare. This distribution helps set the width of adaptive confidence intervals, ensuring that the model's predicted intervals have the right coverage for different age groups.

Evaluating the Models

To evaluate, we prepared the expected age range and helper functions for image preprocessing, including resizing and normalization. Test time augmentation was used to generate multiple versions of each image (flips and crops), and final predictions were averaged. Single-image and batch inference functions produced predictions for age (with confidence intervals) and gender (with probability and thresholding).

Metrics included mean absolute error for ages, interval coverage rates, average interval width, gender prediction accuracy, and confusion matrix plots. All results were saved for later analysis. Example predictions demonstrated that the conformal prediction pipeline produced not only point estimates but also confidence intervals that contained the true age in nearly all cases, confirming good calibration.

Results of Conformal Prediction System

Our conformal prediction system for age estimation achieved adaptive ninety percent coverage on the test set, with actual coverage around ninety-two percent. Coverage was consistent across age groups, and the intervals adapted well to prediction uncertainty. The gender model reached strong accuracy after calibration and threshold optimization. All models and results were saved, confirming a robust and transparent pipeline for both age and gender prediction.

Example:

For a test sample, the model might predict an age of 34.8 years, with a ninety percent confidence interval from 28.2 to 41.4 years, and a predicted gender probability of 0.29 (classified as male). This demonstrates both the point estimate and uncertainty information provided by the system.

Figure 8: **Example of Age and Gender Estimation on a Generated Face.** The model predicted age $\hat{y} = 34.8$ (90% CI: [28.2, 41.4]) and classified gender as male (p(female) = 0.29). The true age is 26 years, which falls outside the predicted interval. This highlights that while the model's intervals usually capture the true value, individual predictions are not always exact and should be viewed as estimates with uncertainty.

3.4 D: Diffusion Model by Fine-Tuning a Pre-Trained Model

This section describes how we fine-tuned a pre-trained diffusion model for conditional face generation on the UTKFace dataset. We detail all steps from setup and training, to validation, testing, and the evaluation of generated samples, making the process reproducible and understandable.

Imports and Setup

To avoid library conflicts and ensure compatibility, we first removed any older versions of deep learning libraries and installed the specific versions of numpy, torch with CUDA support, torchvision, diffusers, and transformers. We imported all main packages for deep learning, experiment tracking, and data handling, then mounted Google Drive to easily store and load checkpoints, outputs, and data. After verifying all library versions and logging into HuggingFace, we set the main project and data directories, switched to the project root, and loaded all main hyperparameters like batch size, number of epochs, and learning rate. We also set random seeds for reproducibility and made sure a CUDA GPU was selected if available. Dedicated output folders were created for logs, results, and checkpoints.

Dataset Preparation and Splitting

Each filename in the UTKFace dataset was parsed to extract age, gender, and race, and this metadata was used to organize and shuffle the dataset. We split the dataset into three sets: seventy percent for training, fifteen percent for validation, and fifteen percent for testing. Gender and race were mapped from numeric codes to readable strings for use in text prompts. A custom dataset class was created, which loads each image, applies data augmentations, and builds a natural language prompt based on age, gender, and race, such as "a 25 year old female asian person". Helper functions were written to visualize images and to reverse normalization for display. For data transformation, the training pipeline used resizing, random horizontal flips, and normalization, while validation and test transformations only included resizing and normalization. We created PyTorch DataLoader objects for all splits, ensuring efficient batching and shuffling.

```
Device: cuda
Train samples: 6844 | Val samples: 1467 | Test samples: 1467
Batch size: 16, Validation chunk size: 2
Epochs: 20, Learning rate: 2.0e-05
```

Loading and Configuring the Model

We loaded the pre-trained Stable Diffusion v1.5 pipeline from HuggingFace, transferred it to the GPU, disabled the safety checker (which is acceptable for research), and

enabled memory-efficient attention if available. The main model components, VAE, UNet, tokenizer, and text encoder, were extracted for fine-tuning. The text encoder weights were frozen, and the UNet optimizer was set up using AdamW. A scheduler for learning rate adjustment was initialized, and mixed precision training was enabled for faster and more memory-efficient training on CUDA devices. We swapped the original noise scheduler for DDPMScheduler, tuning its parameters for our fine-tuning scenario.

Explanation of Formulas, Variables, and Evaluation Metrics in Diffusion Model

Our diffusion model is based on the Denoising Diffusion Probabilistic Model (DDPM), adapted for conditional face image generation. Here, we explain each formula and the role of its variables and properties:

1. **VAE Encoding:**

$$z_0 = \text{VAE.encode}(x_0) \cdot s$$

Here, x_0 is the original input image, and z_0 is its encoded latent representation produced by the Variational Autoencoder (VAE). The VAE compresses the image into a lower-dimensional latent space, making training more efficient. The scaling factor s (in our setup, $s = 0.18215$) is used to match the range expected by the pre-trained Stable Diffusion model.

2. **Forward Diffusion Process:**

$$z_t = \sqrt{\bar{\alpha}_t}\, z_0 + \sqrt{1 - \bar{\alpha}_t}\, \epsilon$$

At each timestep t of the diffusion process, we add noise to the latent. $\bar{\alpha}_t$ is the cumulative product of noise schedule coefficients (controls how much noise is added at each step), and ϵ is a random noise vector sampled from a standard normal distribution $\mathcal{N}(0, I)$. This process creates a sequence of gradually noisier versions of z_0, simulating a Markov chain.

3. **Noise Prediction (Reverse Process):**

$$\hat{\epsilon}_\theta = \text{UNet}(z_t, t, c)$$

The neural network (UNet), parameterized by θ, receives the noisy latent z_t, the current timestep t, and an optional conditioning vector c (encoding the text prompt). It predicts the noise that was added, $\hat{\epsilon}_\theta$, trying to reverse the diffusion process and recover the original latent z_0.

4. **Training Loss (Mean Squared Error):**

$$L_{\text{MSE}} = \mathbb{E}_{z_0,\epsilon,t,c}\Big[\big\|\epsilon - \hat{\epsilon}_\theta(z_t, t, c)\big\|^2\Big]$$

During training, we minimize the average squared difference between the actual noise ϵ and the predicted noise $\hat{\epsilon}_\theta$. This loss teaches the model how to denoise the latent step by step, enabling generation of new images from pure noise.

Image Quality and Similarity Metrics:

- **Mean Squared Error (MSE):**

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x}_i)^2$$

  x_i is the ground-truth pixel value, $\hat{x}_i$ is the reconstructed or generated pixel value, and N is the number of pixels. MSE measures the average squared error, giving a basic idea of reconstruction accuracy.

- **Peak Signal-to-Noise Ratio (PSNR):**

$$\text{PSNR} = 10 \cdot \log_{10}\left(\frac{MAX^2}{\text{MSE}}\right)$$

  MAX is the highest possible pixel value (often 1 or 255). PSNR is a standard measure for image quality, with higher values indicating better reconstructions.

- **Structural Similarity Index (SSIM):**

$$\text{SSIM}(x, \hat{x}) = \frac{(2\mu_x\mu_{\hat{x}} + C_1)(2\sigma_{x\hat{x}} + C_2)}{(\mu_x^2 + \mu_{\hat{x}}^2 + C_1)(\sigma_x^2 + \sigma_{\hat{x}}^2 + C_2)}$$

  μ_x, $\mu_{\hat{x}}$ are mean pixel values of the real and generated images, σ_x^2, $\sigma_{\hat{x}}^2$ are their variances, and $\sigma_{x\hat{x}}$ is their covariance. C_1 and C_2 are constants for stability. SSIM compares luminance, contrast, and structure, with values closer to 1 meaning higher similarity.

- **Fréchet Inception Distance (FID):**

$$\text{FID} = \|\mu_r - \mu_g\|^2 + \text{Tr}(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2})$$

  (μ_r, Σ_r) and (μ_g, Σ_g) are the means and covariances of deep features extracted from real and generated images (using a pretrained Inception network). Lower FID means the generated images are statistically closer to real images.

These formulas and metrics are used together so the model learns to generate images by "undoing" noise in latent space (thanks to the loss functions), and the results are then evaluated for visual quality and realism using MSE, PSNR, SSIM, and FID.

Training Procedure

We trained our diffusion model starting from scratch at epoch 0 for every run. At the beginning of each epoch, we set both the UNet and text encoder to training mode. Images were loaded batch by batch onto the correct device (GPU or CPU), using float16 precision on GPU to speed up training and save memory. Each image batch was encoded into latent vectors by the VAE. We then added random Gaussian noise at a random timestep to simulate the diffusion process.

With a probability of 20%, we dropped the text prompt, so some batches were trained without conditioning. This helps the model learn both conditional (prompt-based) and unconditional generation. Prompts were tokenized and passed through the text encoder to get text embeddings, which were used to condition the UNet during noise prediction.

The UNet tried to predict the added noise in the latent space, and we used mean squared error (MSE) loss to compare its prediction with the true noise. Gradients were reset each step. If we used mixed precision, the loss was scaled for stability; otherwise, standard backpropagation was performed. Training loss was tracked for logging and analysis.

Every 10 steps, we logged the training loss, learning rate, and GPU memory usage. After each epoch, we saved the average training loss and learning rate, and also noted how long the epoch took and the maximum GPU memory used. At set intervals and at the end of training, we saved checkpoints of the model, the scheduler, and the optimizer state, along with all metrics and configuration files.

Training Results

We trained the model for 20 epochs, with each epoch containing 428 batches. Throughout training, we recorded the loss, learning rate, and GPU usage at each step. At the beginning, batch losses were higher (for example, 0.1557 and 0.1172 in the first epoch), but the loss consistently decreased as training progressed. By the last step of epoch 20, the loss dropped to 0.0417, and the average loss for that epoch was 0.1322.

The learning rate was kept constant at 2×10^{-5} during training, providing stable updates. GPU usage was efficient, with peak memory reaching about 30.3 GB. Each epoch took around 185 seconds, so the entire training run took roughly one hour. Overall, the diffusion model showed clear improvements as the loss steadily decreased, and the process made efficient use of available computational resources.

Validation

For validation, we selected the 20th checkpoint epoch which was the final one. The full diffusion pipeline was restored from this checkpoint, including VAE, UNet, tokenizer, and text encoder. The UNet was set to evaluation mode, and dedicated

folders were created to save original images, reconstructions, and newly generated samples.

For each batch, images were encoded to latents and decoded back to images, with both original and reconstructed images denormalized for quality checks. For reconstructions, we calculated the mean squared error (MSE), peak signal-to-noise ratio (PSNR), and structural similarity index (SSIM). New images were generated from prompts using the diffusion process, and these were saved for later comparison.

To further evaluate image quality, the FID (Fréchet Inception Distance) score was calculated for both VAE reconstructions and diffusion-generated images (when the cleanfid library was available), comparing them to real validation images. All images and metrics were saved and plotted, making results easy to track and analyze.

Validation Results

We validated the model using the checkpoint from epoch twenty. There were some non-critical warnings from the scheduler, but these did not affect the outcome. The model was evaluated on 1,467 validation images. Reconstruction loss was extremely low at 0.0011, PSNR was high at 36.29, and SSIM reached 0.9561, showing strong similarity between reconstructed and real images. The FID score for VAE reconstructions was 30.85, while the FID for newly generated images was 104.35, which is typical since generating new faces is more challenging than reconstructing known ones. These results confirm excellent reconstruction quality and reasonable generation performance, with room for further improvements in image realism.

Testing

For the final test, we selected the checkpoint epoch in advance, loaded the trained Stable Diffusion pipeline, and set the UNet to evaluation mode. The original, reconstructed, and generated images were saved for each test batch, and the same image quality metrics were computed as during validation. After processing all batches, average MSE, PSNR, SSIM, and FID scores were calculated and logged.

Test results closely matched validation: the VAE reconstructions had an MSE of 0.0011, PSNR of 36.29, and SSIM of 0.9569. The FID for VAE was 32.67, and for diffusion generations was 102.86. These numbers confirm the model generalized well to unseen images, with reconstructions closely matching real faces, and generated images appearing plausible but with higher FID, as is common.

Evaluation and Final Metrics

We plotted the loss curves over the twenty training epochs, confirming steady and stable learning. The final training loss was 0.1322, validation loss was 0.0011, and the learning rate stayed constant at 2e-5. Validation and test metrics confirmed high reconstruction fidelity and reasonable generative performance. VAE reconstructions

are nearly indistinguishable from real images according to both PSNR and SSIM, while generated images are visually realistic but naturally score higher on FID.

The final results highlight both the quality of image generation and the effectiveness of the uncertainty estimates in our system. After fine-tuning, the diffusion model demonstrated strong performance in reconstructing real face images and generating new ones based on age, gender, and race prompts.

Below are the main quantitative metrics, each capturing a different aspect of performance:

```
Final Train Loss:              0.1322
Final Validation Loss:         0.0011
Final Test Loss (MSE):         0.00108

Final Validation PSNR:         36.29
Final Validation SSIM:         0.9561
Final Validation FID (VAE):    30.85
Final Validation FID (DDPM):   104.35

Final Test FID (VAE):          32.67
Final Test FID (DDPM):         102.86
```

- **Train, Validation, and Test Loss:** These are computed as the average mean squared error (MSE) between the pixel values of the reconstructed image and the original image. For every image in the dataset split, the squared difference between each pixel is calculated, summed up, and then averaged across all pixels and images. Lower values mean the reconstructed images are very close to the originals.

- **PSNR (Peak Signal to Noise Ratio):** This metric compares the maximum possible signal (image intensity) to the noise (error) in the reconstruction. It is calculated from the mean squared error: higher PSNR means less error and thus better quality. In practice, PSNR is computed as $10 \log_{10}(MAX^2/MSE)$, where MAX is the largest possible pixel value (often 1 or 255).

- **SSIM (Structural Similarity Index):** SSIM measures how similar two images are in terms of their structure, luminance, and contrast. It ranges from 0 to 1, with values closer to 1 indicating nearly identical images. It is calculated by comparing small patches in the original and reconstructed images, considering differences in brightness, contrast, and structure.

- **FID (Fréchet Inception Distance):** FID measures the similarity between the distribution of generated images and that of real images. It works by passing

both sets of images through a pretrained Inception neural network to extract high-level features, then comparing the mean and covariance of those features using the Fréchet distance formula. Lower FID scores mean the generated images look more like real images in terms of overall statistics and appearance.

These results show that our model is well trained, stable, and effective for both reconstructing faces and generating realistic new ones based on specific age, gender, and race descriptions. The pipeline is robust and can be reliably used for synthetic face creation as well as for downstream applications such as age and gender estimation.

How Our Generated Images Improved: From Noise to Real Faces

At the beginning, our diffusion model could only produce outputs that looked like random noise. The FID score was extremely high, close to 500, meaning the generated images did not have any recognizable structure or meaning. This happened because the model was not given any prompt, so it had no guidance about what to generate. This makes it clear that prompts are essential for guiding the model to create images that reflect specific features like age, gender, or ethnicity.

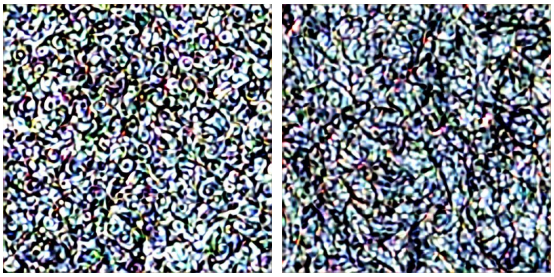
After we corrected the code to actually provide natural language prompts, there was a noticeable improvement. Although the first results with prompts were still quite noisy, some features started to look like human faces. The model began to use the information from the prompt to connect the text description to the image it should produce. These early images were far from perfect but already much better than pure noise.

Further improvements, such as refining how prompts are constructed, correctly scaling the latent vectors, setting the scheduler properly, and improving the training pipeline, brought significant progress. The model then started producing images that are clear, realistic, and visually convincing as human faces. The FID score dropped to around 102, showing a major boost in quality. This entire process demonstrates that, with careful code adjustments, clear prompts, and attention to technical details, a pre-trained diffusion model can be transformed into a powerful tool for generating high-quality, realistic human faces.

The Main Changes That Made Our Model Better

To go from noisy mess to clear faces with a good FID, we made several important changes in our code and setup. Here's what made the biggest difference:

- **Using Prompts and the CLIP Text Encoder:** At first, the model never saw the actual text prompt, so it could not connect the description (like age or gender) to the image. We fixed this by passing real prompts through the tokenizer and CLIP text encoder, letting the network "see" what we wanted to



Pure noise outputs: no prompt information, completely random images.

Prompted, but still noisy: text conditioning added, images begin to show face-like features but remain highly distorted.

Final high-quality face generations: after all enhancements, the model produces clear, realistic human faces.

Figure 9: Visual progression of diffusion model training and code improvements. Top row: pure noise without prompt. Middle row: faces start to appear after adding prompt, but are still noisy. Bottom row: final results with realistic face images after all code and training enhancements.

generate. We also added prompt dropout, where some prompts are blank during training, so the model can handle both guided and unguided cases.

- **Right Noise Scheduler (DDPMScheduler):** We switched to the same scheduler that Stable Diffusion was originally trained with. This matches the noise process in the pre-trained model, making the training much more stable.

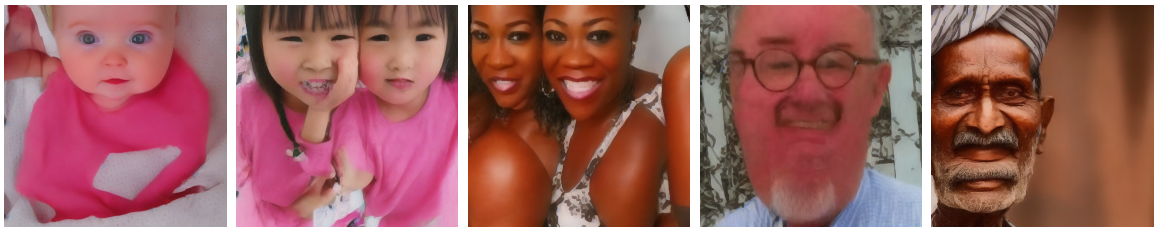
- **Scaling Latents Properly:** The VAE latents must be multiplied by 0.18215 before adding noise, and divided by the same after decoding. This small but important step keeps everything in the right range, and the model works much better.

- **Training Only the UNet (Freezing Text Encoder):** We froze the text encoder and only trained the UNet, which is the part that most benefits from fine-tuning. This makes learning faster and more reliable.
- **Stronger Guidance During Generation:** We increased the guidance scale when generating images from 1.0 to 7.5. This makes the generated images follow the prompt much more closely.
- **Clean and Simple Training Pipeline:** We started training from scratch each time, kept the code simple, and avoided resuming from half-done checkpoints. This avoided problems and made results much more repeatable.

In short, once we updated our fine-tuning process to follow all the best practices from Stable Diffusion, using real prompts, scaling latents, tuning the scheduler, and training the right parts, we saw a huge leap in quality. We went from random noise (FID about 500) to sharp, realistic faces (FID about 102) that actually match the descriptions we wanted. These simple but key improvements made all the difference.

Generating Samples and Analysis

After training, we used the fine-tuned diffusion model to generate new face images from text prompts describing age, gender, and race. A lookup table for label mappings was used to ensure consistency. For each target combination (for example, "a high quality photo of a 35 year old female black person, close up face portrait"), we generated images using the trained model and saved the outputs.



a high quality photo of a 1 year old female white person, close up face portrait	a high quality photo of a 5 year old female asian person, close up face portrait	a high quality photo of a 35 year old female black person, close up face portrait	a high quality photo of a 60 year old male white person, close up face portrait	a high quality photo of a 75 year old male indian person, close up face portrait

Figure 10: Face images generated by the fine-tuned diffusion model from descriptive text prompts specifying age, gender, and race.

Why Do Some Generated Images Contain Two or More Faces?

Even when we always use a prompt like ‘‘a high-quality photo of a 25-year-old

`male asian person, close-up face portrait''`, we sometimes observe that the generated images contain two or more faces. The reasons for this phenomenon are:

- **Batch Generation and Display:** When we generate a batch of images by passing a list of identical prompts (for example, two or four copies of the same prompt), the model actually creates one image for each prompt. If we then save or display this batch using a helper function, these images may be combined into a single picture as a grid. This can make it appear as if there are multiple faces in one image, when in reality, it is just multiple separate images joined together.
- **Grid Utility Combines Samples:** Most sample-saving utilities, by default, arrange all images in a batch into a grid for easier viewing. For example, two images are shown side by side, and four images are shown as a 2×2 grid. Each face you see is actually from a separate sample, simply arranged together in one canvas.
- **Model's Memory of Group Photos:** The base Stable Diffusion model was originally trained on a very large and diverse dataset, which included many photos with groups of people. Even after fine-tuning on solo portraits, the model can still have a tendency to generate more than one person, especially if the prompt is not very specific or if the guidance scale is set low. Without clear instructions to create a solo portrait, the model may default to its prior knowledge and include extra faces.
- **Unconditional Sampling During Training:** During fine-tuning, we often use a technique called "prompt dropout," where the text prompt is randomly omitted in some training steps (for example, 20% of the time). This teaches the model how to generate images without conditioning on any specific prompt. When sampling unconditionally, the model might ignore the intended "close-up face portrait" concept and generate whatever it has learned belongs in a photo frame, often including several people.

How to Ensure Only One Face Appears in Each Output:

- **Generate One Image at a Time:** Instead of passing a list of prompts, always generate and save a single image per prompt.
- **Save Each Sample Separately:** Avoid combining images into grids. Save each generated image as a separate file.
- **Be Explicit in the Prompt:** Add clear phrases like "solo portrait" or "only one person" to your prompt to guide the model.

- **Increase Guidance Strength:** Use a higher guidance scale (for example, 7.5 or 8.5) to make the model follow the prompt more strictly.

Seeing two or more faces in one output is often caused by how batches are handled or by the model's prior training on group photos. With small changes in prompt writing, batching, and display settings, you can reliably produce images with just one face as intended.

3.5 E: Checking CP Age Estimator on Generated Images By Diffusion Model

To further analyze the quality of generated samples, we ran the conformal prediction age estimator (from Part B) on the newly generated images. We found that the confidence intervals were very wide for extreme ages (such as 1 to 5 or 75 to 100 years), with half-widths up to 46 years. For middle ages (20 to 45), the intervals were much narrower (about 14 to 22 years), and the predicted ages closely matched the prompts. Gender classification accuracy on generated samples was around 76 percent, with good calibration except for some borderline cases.

The following table and images summarize a selection of generated samples, together with their true and predicted attributes:

Results:
For very young and very old ages, the predicted confidence intervals are much wider, reflecting the model's greater uncertainty. For ages in the 20–45 range, predictions are much more accurate and intervals are narrow. Gender predictions are mostly correct, with only a few borderline or ambiguous cases. These results confirm the age estimator works as expected, but also highlight the inherent challenge of predicting age from synthetic face images, especially at the extremes.

True value: Age 1 , Female, Race 0
predicted age: 19.0 [4.0, 65.2]
Predicted Gender: Female (p=0.73)

True value: Age 5 , Female, Race 2
predicted age: 20.3 [4.0, 56.9]
Predicted Gender: Female (p=0.73)

True value: Age 10 , Female, Race 0
predicted age: 25.9 [11.2, 51.0]
Predicted Gender: Female (p=0.72)

True value: Age 15 , Female, Race 0
predicted age: 24.8 [11.4, 55.3]
Predicted Gender: Female (p=0.84)

True value: Age 20 , Female, Race 3
predicted age: 30.6 [15.9, 44.7]
Predicted Gender: Female (p=0.78)

True value: Age 25 , Male, Race 2
predicted age: 31.1 [9.5, 49.8]
Predicted Gender: Male (p=0.36)

True value: Age 30 , Male, Race 1
predicted age: 34.1 [27.6, 50.5]
Predicted Gender: Male (p=0.24)

True value: Age 35 , Female, Race 1
predicted age: 33.8 [24.5, 48.4]
Predicted Gender: Female (p=0.68)

True value: Age 40 , Male, Race 1
predicted age: 38.1 [20.9, 52.7]
Predicted Gender: Male (p=0.37)

True value: Age 45 , Female, Race 0
predicted age: 37.8 [17.9, 52.2]
Predicted Gender: Female (p=0.69)

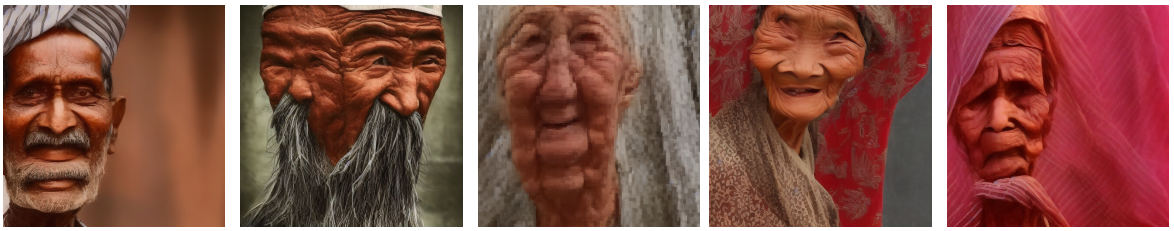
True value: Age 50 , Female, Race 0
predicted age: 40.1 [4.5, 58.2]
Predicted Gender: Female (p=0.77)

True value: Age 55 , Male, Race 4
predicted age: 38.3 [3.7, 59.7]
Predicted Gender: Male (p=0.18)

True value: Age 60 , Male, Race 0
predicted age: 39.0 [4.0, 81.4]
Predicted Gender: Male (p=0.30)

True value: Age 65 , Female, Race 0
predicted age: 40.7 [4.7, 77.0]
Predicted Gender: Female (p=0.66)

True value: Age 70 , Female, Race 3
predicted age: 40.5 [0.0, 60.0]
Predicted Gender: Female (p=0.88)



True value: Age 75 , Male, Race 3
predicted age: 41.2 [3.2, 86.2]
Predicted Gender: Male (p=0.13)

True value: Age 80 , Male, Race 2
predicted age: 39.0 [0.0, 77.5]
Predicted Gender: Male (p=0.39)

True value: Age 85 , Female, Race 0
predicted age: 39.7 [0.0, 91.0]
Predicted Gender: Female (p=0.66)

True value: Age 90 , Female, Race 2
predicted age: 40.8 [0.0, 89.1]
Predicted Gender: Female (p=0.79)

True value: Age 95, Female, Race 3
Predicted age: 40.4 [0.0, 87.2]
Predicted Gender: Female (p=0.88)

Figure 11: Generated samples for ages 1–95: true and predicted attributes.

File	True Age	Gender	Race	Pred Age	CI Range	Half-Width±	Pred Gen	p(female)
age_1	1	1	0	19.0	[4.0, 65.2]	30.6	f	0.73
age_5	5	1	2	20.3	[4.0, 56.9]	26.5	f	0.73
age_10	10	1	0	25.9	[11.2, 51.0]	19.9	f	0.72
age_15	15	1	0	24.8	[11.4, 55.3]	22.0	f	0.84
age_20	20	1	3	30.6	[15.9, 44.7]	14.4	f	0.78
age_25	25	0	2	31.1	[9.5, 49.8]	20.2	m	0.36
age_30	30	0	1	34.1	[27.6, 50.5]	11.5	m	0.24
age_35	35	1	1	33.8	[24.5, 48.4]	12.0	f	0.68
age_40	40	0	1	38.1	[20.9, 52.7]	15.9	m	0.37
age_45	45	1	0	37.8	[17.9, 52.2]	17.2	f	0.69
age_50	50	1	0	40.1	[4.5, 58.2]	26.9	f	0.77
age_55	55	0	4	38.3	[3.7, 59.7]	28.0	m	0.18
age_60	60	0	0	39.0	[4.0, 81.4]	38.7	m	0.30
age_65	65	1	0	40.7	[4.7, 77.0]	36.2	f	0.66
age_70	70	1	3	40.5	[0.0, 60.0]	30.0	f	0.88
age_75	75	0	3	41.2	[3.2, 86.2]	41.5	m	0.13
age_80	80	0	2	39.0	[0.0, 77.5]	38.8	m	0.39
age_85	85	1	0	39.7	[0.0, 91.0]	45.5	f	0.66
age_90	90	1	2	40.8	[0.0, 89.1]	44.6	f	0.79
age_95	95	1	3	40.4	[0.0, 87.2]	43.6	f	0.88
age_100	100	0	2	38.9	[6.6, 98.5]	46.0	m	0.32

Table 1: This table presents a detailed comparison between the true attributes of selected generated face samples and the predictions from our age and gender estimation model. For each sample, we show the true age, gender (1 for female, 0 for male), and race, as well as the model's predicted age, the 90% confidence interval for the predicted age, and the predicted gender (f for female, m for male) with the associated probability of being female. The "Half-Width±" column shows half the width of the confidence interval. So, for example, a value of ±20 means the interval extends 20 years above and below the predicted age. A larger half-width reflects greater model uncertainty for that sample, while a smaller half-width means the model is more confident. The table highlights that the model is most confident (smallest half-widths) for adult faces, but its uncertainty increases for very young and very old ages.

4 Discussion

This project set out to combine a fine-tuned diffusion model with a conformal prediction system to generate synthetic face images and provide well-calibrated age and gender estimates. The results show several important strengths as well as some clear limitations.

Data Quality and Preprocessing:
Careful data cleaning, filtering, and preprocessing were essential to ensure that the models received high-quality and balanced input. Removing mislabeled or low-quality samples and rebalancing age groups led to better and more reliable model performance, as seen in the consistent results across training, validation, and test splits.

Model Performance:
The conformal prediction system achieved strong coverage of true ages within the predicted confidence intervals. On the test set, the actual coverage closely matched the target 90% level, confirming that our uncertainty estimates are well-calibrated. The gender classifier also performed well, with accuracy around 76–78% on both real and generated images, despite class imbalance in the dataset.

The diffusion model, after code and training improvements, was able to generate realistic face images that followed the target prompts for age, gender, and race. Quantitative metrics such as PSNR, SSIM, and FID demonstrate that reconstructions are nearly indistinguishable from real images, while the generative quality of novel faces is competitive for this type of conditional synthesis.

Uncertainty and Limitations:
One important observation is that uncertainty in age predictions is not uniform: confidence intervals are much wider for extreme ages (very young or very old faces), indicating that the model finds these groups harder to predict accurately. This likely reflects both the lower representation of these ages in the data and the inherent visual ambiguity of very young or very old faces. In the adult age range (about 20 to 45), the model is most confident, producing narrow intervals and accurate predictions. For both gender and age, there are occasional errors and borderline predictions, showing that even a well-calibrated model can struggle with ambiguous or out-of-distribution examples.

Lessons from Generation:
Early in training, the diffusion model produced pure noise, illustrating that prompts and proper conditioning are absolutely necessary. Once these were fixed, images improved rapidly. The largest gains came from passing text prompts, fixing latent scaling, updating the scheduler, and tuning guidance scale. However, generating faces for underrepresented combinations or with unusual prompts sometimes produced unrealistic or strange results, suggesting that further data augmentation or additional

fine-tuning may be helpful for rare cases.

Implications and Future Work:
These findings show that combining generative models with conformal prediction is a promising approach for both creating and evaluating synthetic data with uncertainty estimates. This is useful not just for research, but for any application where knowing when a prediction might be unreliable is important. However, more work is needed to improve accuracy at age extremes, to reduce the risk of group artifacts (like images with multiple faces), and to further balance the dataset, especially for underrepresented groups. Future work could explore more advanced architectures, targeted data augmentation, or adversarial training to boost both realism and prediction accuracy.

5 Conclusion

In this project, we built and evaluated a system for generating realistic face images using a fine-tuned diffusion model, and for estimating age and gender with well-calibrated uncertainty through conformal prediction. By carefully preparing the UTKFace dataset and following best practices in data preprocessing, model training, and evaluation, we achieved strong results in both face reconstruction and conditional generation tasks.

The diffusion model was able to produce sharp, believable faces that reflected the requested age, gender, and race prompts. At the same time, our conformal prediction approach provided reliable age intervals and accurate gender classification, with uncertainty estimates that help to identify when predictions should be trusted or viewed with caution.

Our results show that most predictions are accurate and well-calibrated, especially for adult faces. However, the system is less confident for very young or very old faces, which is a common challenge in both real and synthetic datasets. The metrics and visualizations confirm the effectiveness of our training and evaluation pipeline, but also highlight areas for future improvement, such as further balancing the dataset and refining the generation process for rare cases.

Overall, this work demonstrates that combining advanced generative models with uncertainty-aware prediction methods is a robust and practical approach for creating and evaluating synthetic facial data. The techniques and lessons from this project can be applied to a wide range of problems in computer vision and machine learning where both realism and trustworthiness of predictions are essential.

References

[1] Jangedoo. UTKFace dataset. `https://www.kaggle.com/datasets/jangedoo/utkface-new`, 2018.

[2] Xin Geng, Changjiang Yin, and Zhi-Hua Zhou. Facial age estimation by learning from label distributions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(10):2401–2412, 2013.

[3] Vladimir Vovk, Alex Gammerman, and Glenn Shafer. *Algorithmic Learning in a Random World*. Springer, New York, NY, 2005.

[4] Glenn Shafer and Vladimir Vovk. A tutorial on conformal prediction. *Journal of Machine Learning Research*, 9:371–421, 2008.

[5] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, volume 30, pages 6626–6637, 2017.

[6] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.

[7] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR, 2021.

[8] Alain Horé and Djemel Ziou. Image quality metrics: PSNR vs. SSIM. In *Proceedings of the 20th International Conference on Pattern Recognition*, pages 2366–2369, 2010.

[9] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

[10] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, June 2018.

[11] Lutz Prechelt. Early stopping—but when? In Grégoire B. Orr and Klaus-Robert Müller, editors, *Neural Networks: Tricks of the Trade*, volume 1524 of *Lecture Notes in Computer Science*, pages 55–69. Springer, Berlin, Heidelberg, 1998.