Diffusion-Based Generative Age Estimation with Conformal Prediction

Statistical Learning - Final Project

Professors: Prof. Pierpaolo Brutti

Student:

Name: Arman
Surname: Feili
Matricola: 2101835
Email: feili.2101835@studenti.uniroma1.it

SAPIENZA
UNIVERSITÀ DI ROMA

Project Goal and Motivation

Goal:
Develop and test a complete system for age estimation from facial images by combining two powerful methods:

- **Conformal Prediction:** Produces not just a single age guess, but a confidence interval, so we know how sure the model is.
- **Diffusion Model:** Generates highly realistic synthetic faces from simple text prompts describing age, gender, and ethnicity.

Why this project?

- Age estimation from faces is useful in areas like demographic research, user experience, and security.
- Most existing models only give a point estimate, without telling us how confident they are.
- Conformal prediction solves this by wrapping any model to provide reliable confidence intervals.
- Advances in generative AI (like diffusion models) now let us create realistic fake faces from text, great for testing age predictors, and for exploring bias or limits of prediction systems.

The question we try to answer:

- **Can an age prediction system remain accurate and trustworthy when faced with fully generated synthetic faces, not just real ones?**

Challenges & Limitations

Compute Power:

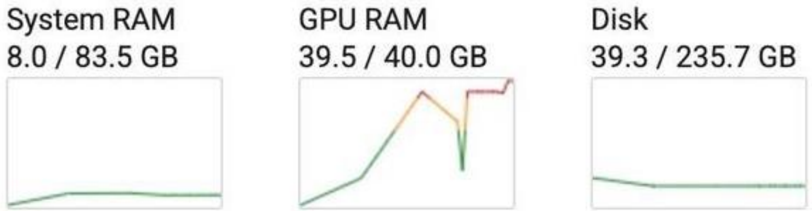
- We did not have access to a high-end Nvidia GPU on our local machines.
- Most model training and testing had to be done on **Google Colab**, where we purchased Colab Pro three separate times and used up about **250 compute units** to access A100 GPUs.
- Less powerful GPUs (like T4 or L4) were often too weak, even loading large models or full-size images would fail or crash sessions.

Frequent Technical Issues:

- Running out of GPU memory or having **Colab sessions crash** was common, especially when scaling from a small debug dataset to the full dataset.
- We often had to debug and **develop code very quickly** to make the most of our compute time, then rerun everything from scratch.
- Fine-tuning both the diffusion and conformal prediction models with different settings was time-consuming and burned through our compute units quickly.

Quality of Generated Images and Unstable Early Results for CP Aged estimation:

- Early in the project, the diffusion model produced only **unrealistic or distorted faces**, even after using a lot of compute.
- It took many rounds of troubleshooting, adjusting prompts, and refining model parameters before we could generate convincing, realistic faces.
- The **conformal prediction** model, when trained locally, initially gave completely nonsensical results (like ages of **450 years**).
- It was unclear whether these issues were from under-training, bugs in the code, or resource limitations.
- Only after moving everything to Colab and carefully refining our pipeline did the models begin producing reasonable predictions.



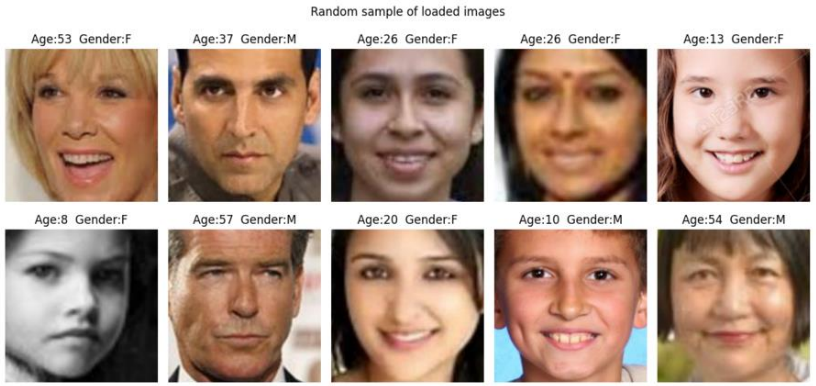
Dataset

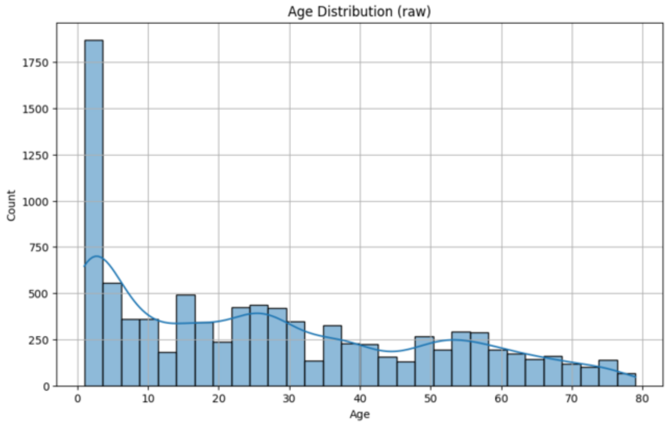
We used the **UTKFace dataset**, which contains over 20,000 natural face images, each labeled with:

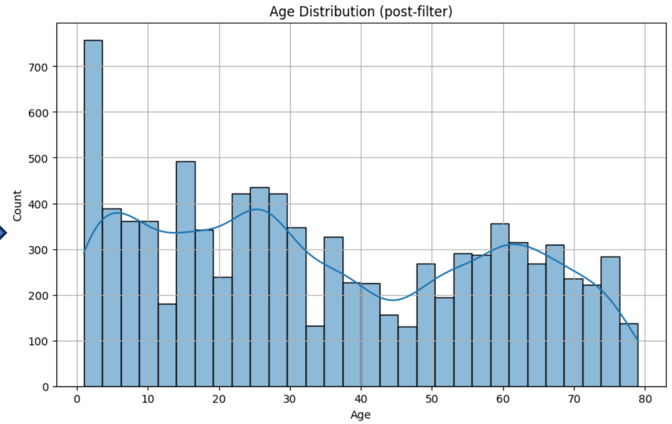
- **Age** (0 to 116 years), **Gender:** (0 = male, 1 = female), **Race:** 0 to 4 (White, Black, Asian, Indian, Other)

Download & Preparation:

- Dataset was downloaded from Kaggle and extracted from a zip file on Google Drive
- All images and labels were checked for correct format and content
- Invalid files and images with missing or incorrect labels were skipped
- Each image resized to **224×224 pixels** and pixel values normalized to [0, 1]
- Ages and genders stored as NumPy arrays
- Ages converted to soft-label vectors using **Label Distribution Age Encoding (LDAE)**, which helps the model learn age as a probability distribution (not just a single value)
- Too many samples from very young and very old groups were rebalanced:
  - Kept 40% of samples under 4 years old
  - Up-sampled seniors (60+)







Conformal Prediction Model: Creating the Age & Gender Models

Flexible Setup:

- We used flexible **environment variables** to easily switch between **quick debug** runs (with a small dataset) and **full training** (full dataset).
- Installed all required **Python packages** (TensorFlow, Keras, etc.) and **Google Drive** was **mounted** for seamless data access.
- Set **fixed random seeds** for Python, NumPy, and TensorFlow to ensure fully **reproducible** results.

Data Splitting:

- Dataset was split into **training (80%), calibration (20% of train), and test (20%) sets**, and grouped by 10-year age bins and gender
    - Train: 5834 samples (44.6% male, 55.4% female)
    - Calibration: 1459 samples (44.6% male, 55.4% female)
    - Test: 1824 samples (44.6% male, 55.4% female)
- **Gender** labels were reshaped to (N,1) for **binary** classifiers.

Age Regression Model Creation:

- Age is predicted with a **MobileNetV2 model** using **224×224 images** and Label Distribution Age Encoding **(LDAE)** with a **softmax** over 90 bins.
- Two augmentation pipelines were applied:
    - One using Keras layers (translation, rotation, cropping, brightness, contrast)
    - One with TensorFlow (random flips, brightness/contrast).
- **Kullback-Leibler Divergence** is used as the **loss function**, and performance is measured by Mean Absolute Error (MAE).
- In training we applied **early stopping**, **learning rate reduction**, and **checkpoints** based on **validation MAE**.

Gender Classification Model Creation :

- Gender is predicted as a **binary output** (male or female) using a **MobileNetV2 model** with a **single sigmoid neuron**.
- **Binary Cross-Entropy** serves as the **loss function**, and **accuracy** is used as the main **metric**.
- In training we applied **early stopping**, **learning rate scheduling**, and model **checkpoints** based on **validation loss**.

Conformal Prediction Model: Formulas

Residual Calculation:

- Compute the absolute error for each calibration sample using $r_i = |\hat{y}_i - y_i|$, where $\hat{y}_i$ is the predicted age and y_i is the true age.
- This measures how far the model's prediction is from the actual value, giving us the raw prediction error for each sample.
- Used on the calibration set to assess baseline prediction errors.

Normalized Residuals:

- Calculate $\tilde{r}_i = \frac{|\hat{y}_i - y_i|}{\sqrt{\hat{y}_i + 1}}$, dividing the error by the square root of the predicted age plus one.
- This normalization accounts for the fact that predicting ages at the extremes (very young or very old) is more difficult, so it makes interval widths fairer across all age ranges.
- Used to make sure confidence intervals are neither too wide nor too narrow for any specific age group.

Adaptive Quantile Calculation (per age group):

- For each 10-year age group k, calculate $q_{hat,k} = Quantile_{0.9}(\{\hat{r}_i : i \in age\ group\ k\})$, the 90th percentile of the normalized residuals.
- This sets the interval width needed for that age group, ensuring that 90% of predictions in each group are covered by the interval.
- Used to make interval widths specific and adaptive for each age group.

Conformal Prediction Model: Formulas

Adaptive Prediction Interval for Each Test Sample:

- Build a 90% confidence interval for every test prediction:
  - $Lower\ bound = \hat{y} - q_{hat} \cdot \sqrt{\hat{y} + 1}$
  - $Upper\ bound = \hat{y} + q_{hat} \cdot \sqrt{\hat{y} + 1}$
- The interval width scales with prediction difficulty and is customized by age group, so uncertainty is reflected correctly.
- Applied to every test prediction to give a personalized confidence interval for the predicted age.

Final Coverage:

- Calculate the percentage of true ages in the test set that fall inside their predicted intervals.
- To check if our intervals are well-calibrated (aiming for 90% coverage if our method is working as intended).
- Used for model evaluation and to ensure trustworthiness of the predicted confidence intervals.

Key Point:

- Every age prediction comes with a confidence interval that adapts to age and difficulty
- wider for hard cases, tighter for confident ones. This method allows us to provide not just point estimates, but also trustworthy measures of uncertainty for real-world applications.

Conformal Prediction Model: Training

Loss Functions & Augmentation:

- For **gender prediction**, we used **binary cross-entropy loss** with label smoothing, making the model less sensitive to noisy labels.
- For **age prediction**, we used **Kullback-Leibler Divergence (KL-divergence)** as the loss, paired with Label Distribution Age Encoding.
- Class weights were used to fix the imbalance between male and female images. the model gave more weight to male images during training to keep the loss balanced.

Training Steps:

- We started by training the new layers only, keeping the main part of the model (MobileNetV2) frozen.
- We used **early stopping** if validation stopped improving, and lowered the learning rate when needed. The best-performing models were saved automatically.
- Later, we **unfroze the last layers** of MobileNetV2 (except for batch norm layers), used a lower learning rate, and fine-tuned the model.
- Some key formulas:
    - **Binary Cross-Entropy (Gender):**
    - $Loss = -[y\log(p) + (1-y)\log(1-p)]$
    - **KL-Divergence (Age):**
    - $Loss = \sum P_i log\left(\frac{P_i}{Q_i}\right)$, where P_i is the target (LDAE), and Q_i is the predicted output.

Thresholds & Saving Results:

- After training, we made predictions on the calibration set and tested different thresholds for the gender model to find the best F1 score and balanced accuracy.
- We saved the **best thresholds**, model weights, architecture, and full training history (metrics, loss, class weights) for future use.

Conformal Prediction Model: Training Results

Results:

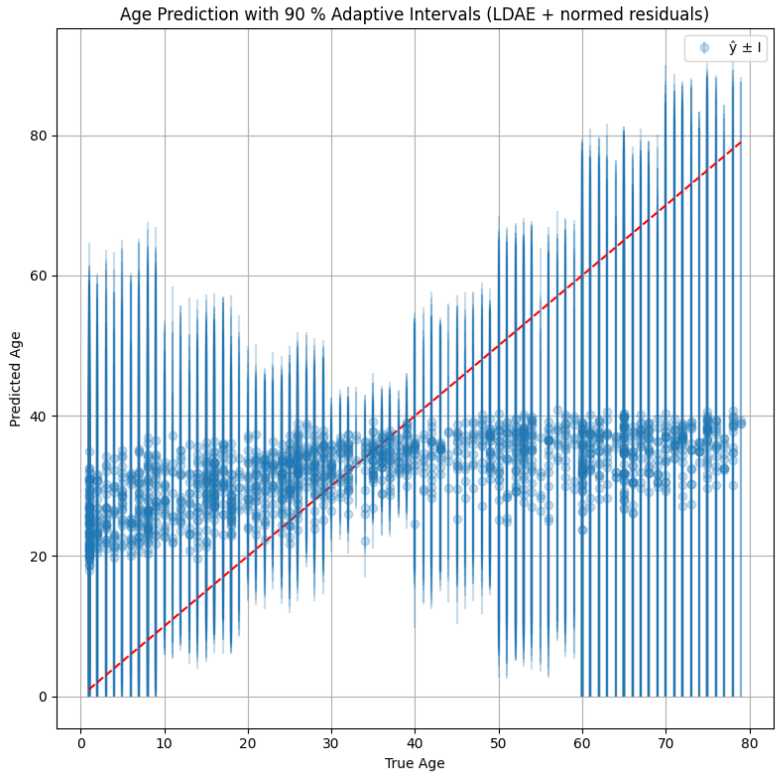
- The **age model** performed consistently with low error throughout training.
- The **gender model** improved steadily and reached about **78% accuracy** after fine-tuning.

Predicted vs. Real Age:

- Each blue dot shows one test image, with the actual age on the x-axis and the predicted age on the y-axis.
- The vertical blue lines on each dot show the **90% confidence range -> how** sure the model is about that prediction.
- The red dashed line represents perfect predictions (where predicted age = real age).
- The model is **less confident** (wider lines) for very young and very old faces, and **more confident** (shorter lines) for ages between **30 and 40**, where predictions are generally more accurate.

How Well Confidence Intervals Work:

- The model correctly included the real age in its 90% confidence range for **91.9%** of test images, very close to the goal of 90%, which means the model is **well-calibrated**.
- This performance is also shown separately for different age groups, so we can see how it behaves for kids, adults, and seniors.
- Even when the predicted age is slightly wrong, it often falls within the model's expected range.
- In short, the model doesn't just guess the age. It also tells us how confident it is, and that confidence is usually accurate.




```
Adaptive 90 % coverage on test set: 0.919

Coverage per age bucket:
00–09: 0.895
10–19: 0.914
20–29: 0.951
30–39: 0.928
40–49: 0.941
50–59: 0.904
60–69: 0.889
70–79: 0.950

Sample predictions (adaptive intervals):
ŷ= 23.6 | I=[  6.5, 40.7] | y=12
ŷ= 31.2 | I=[  2.9, 59.4] | y=8
ŷ= 26.4 | I=[  0.3, 52.5] | y=1
ŷ= 36.9 | I=[ 18.9, 54.9] | y=49
ŷ= 37.2 | I=[ 19.1, 55.2] | y=46
```

Conformal Prediction Model: Evaluation the model

Model Evaluation:

- We measured **age prediction error** using Mean Absolute Error (MAE) and checked how often the true age fell inside the **90% confidence interval** (coverage rate).
- The **width of confidence intervals** tells us how sure the model is:
  - narrower means more confidence, wider means more uncertainty.
- For **gender**, we used overall accuracy and confusion matrix analysis to understand correct and incorrect predictions.
- All results—including predictions, metrics, and confidence ranges—were saved for full transparency.

Key Outcomes & Example:

- The model reached **91.9% coverage**, very close to the intended 90%, showing that its uncertainty estimates are accurate.
- This reliability was consistent across all age groups, and the gender model improved further after threshold tuning.
- **Example**:
  - One prediction was **34.8 years** (range: **28.2–41.4**), real age was **26.** just outside the range.
  - Gender was predicted as male with **29% chance of being female**.
- Overall, the system doesn't just predict. it gives clear, data-driven confidence intervals that help users trust each result.

```
Age ŷ=34.8  CI=[28.2,41.4]
Gender=male  p(female)=0.289 thr=0.5
```


```
{'age_pred': 34.82306914589416,
 'age_interval': (np.float64(28.223387609523066),
  np.float64(41.42275068226525)),
 'gender_prob': 0.2888246774673462,
 'gender_pred': 'male'}
```

Diffusion Model: Fine-Tuning the Pre-Trained Model

Setup and Environment

- Removed old libraries and installed needed ones: numpy, torch (with CUDA), diffusers, and transformers.
- **Mounted Google Drive** for saving/loading checkpoints and logs.
- Set up project folders, fixed random seed for reproducibility, and confirmed **GPU (CUDA)** was available.
- **Logged into HuggingFace** and verified all versions.

Dataset Preparation

- Parsed the UTKFace filenames to extract age, gender, and race, then converted them into readable text for prompts.
- **Split the dataset: 70% training, 15% validation, 15% testing**.
- Created a custom dataset class that loads images, applies transformations, and builds natural language prompts.
- Used simple augmentations for training; only resizing and normalization for validation/testing.

Model Loading and Fine-Tuning

- Loaded **Stable Diffusion v1.5** and moved it to **GPU**. Disabled safety checker and enabled memory-efficient attention.
- **Used VAE, UNet, tokenizer, and text encoder from the model**; froze the text encoder and fine-tuned only the UNet.
- Applied **AdamW optimizer, learning rate scheduler**, and mixed precision for faster training.
- Replaced default noise scheduler with a tuned **DDPMScheduler**.

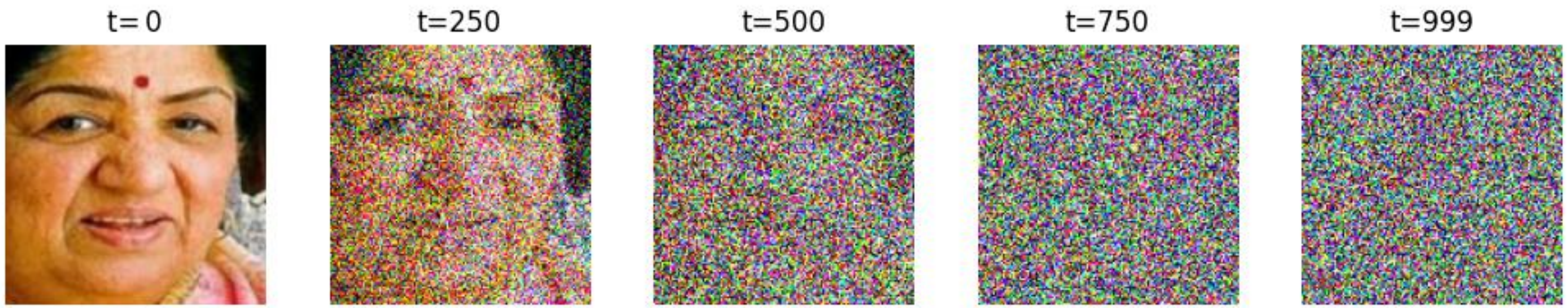
Diffusion Model: Explanation of Variables and their Formulas in Training

VAE Encoding

- We use a **Variational Autoencoder (VAE)** to compress each original input image x_0 into a lower-dimensional latent vector z_0:
  - $z_0 = VAE.encode(x_0) \cdot s$
  - where $s = 0.18215$ is a scaling factor.
- This makes training much more efficient by reducing image dimensionality, but keeps all the necessary features needed for face generation.
- We use this as the very first step in the diffusion model pipeline, before adding noise.

Forward Diffusion Process

- At each timestep t, we generate a noisy version of the latent z_t by mixing the encoded image z_0 with random Gaussian noise ϵ.
  - $z_t = \sqrt{\bar{\alpha}_t} z_0 + \sqrt{1 - \bar{\alpha}_t}\, \epsilon$,
  - where $\bar{\alpha}_t$ determines the amount of noise.
- This process gradually destroys the image, so the model can learn to reverse it and “denoise” step by step during training.
- This is done during both training (to create training pairs) and image generation (as the initial “noise” state).



Diffusion Model: Explanation of Variables and their Formulas in Training

Noise Prediction (Reverse Process)

- The **UNet neural network** takes the noisy latent z_t , timestep t, and condition vector c (from the text prompt) and predicts the noise that was added:

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

  - $\hat{\epsilon}_\theta$ ("epsilon hat sub theta"): The predicted amount of noise that was added to the latent representation (z_t) at timestep t.
  - UNet: The neural network model used for denoising (predicts the noise to remove).
  - z_t : The noisy latent vector of the image at timestep $t.$
  - t: The current timestep in the diffusion process.
  - c: The conditioning vector. (text prompt for age, gender, race)
  - θ: The network's trainable parameters (weights).
- This process teaches the model how to reverse the noise and recover the original image, enabling controlled image synthesis.
- Used in every step of the reverse (denoising) process, both during training and when generating new images.

Training Loss (Mean Squared Error)

- The loss function is the **mean squared error** between the actual noise ϵ and the predicted noise $\hat{\epsilon}_\theta$:

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

  - L_{MSE} : The mean squared error loss used for training..
  - $\mathbb{E}_{z_0,\epsilon,t,c}$: The expectation (average) is taken over all data, noise, timesteps, and conditions.
  - ϵ : The true noise added at step t.
  - $\hat{\epsilon}_\theta(z_t, t, c)$: The noise predicted by UNet for the noisy latent z_t, at timestep t, with conditioning c.
  - $\|\epsilon - \hat{\epsilon}_\theta(z_t, t, c)\|^2$: The squared error between the real and predicted noise.
- Minimizing this loss ensures the UNet learns to accurately denoise and reconstruct realistic images.
- Used as the main loss during training of the diffusion model.

Diffusion Model: Explanation of Evaluation Metrics in Validation and Testing

Mean Squared Error (MSE)

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

- Where:
    - N : Number of pixels in the image.
    - x_i : The value of the i -th pixel in the real (ground-truth) image.
    - $\hat{x}_i$: The value of the i -th pixel in the generated or reconstructed image.
- Calculates the average squared difference between each real pixel value and the corresponding generated pixel.
- Measures how close the generated image is to the ground truth, with lower values indicating better reconstructions.
- Used to evaluate the reconstruction accuracy of generated faces.

Peak Signal-to-Noise Ratio (PSNR)

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

- Where:
    - MAX : The maximum possible pixel value (e.g., 255 for 8-bit images, or 1 for normalized images).
    - MSE : The mean squared error between the real and generated images (as above).
- Compares the maximum possible pixel value to the error, resulting in a score in decibels.
- Higher PSNR means the generated image has less error and is visually clearer.
- Used alongside MSE to assess the quality of generated images.

Diffusion Model: Explanation of Evaluation Metrics in Validation and Testing

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

  - $x, \hat{x}$: The real and generated images, respectively.
  - μ_x : The mean pixel value of x.
  - $\mu_{\hat{x}}$: The mean pixel value of $\hat{x}$.
  - σ_x^2 : The variance of x.
  - $\sigma_{\hat{x}}^2$: The variance of $\hat{x}$.
  - $\sigma_{x\hat{x}}$: The covariance between x and $\hat{x}$.
  - C_1 , C_2 : Small constants for numerical stability (avoid division by zero).
- Measures similarity in luminance, contrast, and structure between the real and generated images.
- Gives a better evaluation of similarity, with values closer to 1 meaning the images are more alike.
- Used as a key metric for evaluating perceptual image quality after generation.

Fréchet Inception Distance (FID): $FID = \left\|\mu_r - \mu_g\right\|^2 + Tr\left(\Sigma_r + \Sigma_g - 2\left(\Sigma_r\Sigma_g\right)^{\frac{1}{2}}\right)$

  - μ_r : The mean of deep features extracted from real images (using a pretrained network).
  - μ_g : The mean of deep features extracted from generated images.
  - Σ_r : The covariance matrix of features from real images.
  - Σ_g : The covariance matrix of features from generated images.
  - Tr : The trace operation (sum of diagonal elements of a matrix).
- Compares the distributions of features (means and covariances) from real and generated images using an Inception network.
- Lower FID scores mean the generated faces are statistically closer to the real ones, capturing both quality and diversity.
- Used to benchmark the overall realism and variety of synthetic faces created by the model.

Diffusion Model: Training

Training Procedure:

1. Training used a CUDA GPU, with:
   - **6844** training samples, **1467** for validation, and **1467** for testing.
   - **Batch size** was **16**, trained for **20 epochs** with a **learning rate** of **2e-5**.

2. Both the **UNet** and **text encoder** were set to training mode at the start of every epoch.
3. For every batch:
   - Images were loaded to the GPU (using float16 precision for efficiency) or CPU if GPU wasn't available.
   - images was **encoded into latent vectors by the VAE** (Variational Autoencoder).
   - At a random timestep, **Gaussian noise** was added to simulate the forward diffusion.

4. To train the model to handle both **conditional** (prompt-based) and **unconditional** generation:
   - In **20% of cases**, the prompt was skipped.
   - Otherwise, the text was tokenized, processed by the **text encoder**, and used to guide the **UNet**'s noise prediction.

5. The **UNet** predicted the added noise at each timestep in latent space, and this was compared to the real noise using **MSE loss**.
6. Before updating weights:
   - Gradients were reset.
   - If using **mixed precision**, the loss was scaled for stable training; otherwise, normal backpropagation was applied.

7. During training:
   - Metrics like **loss**, **learning rate**, and **GPU usage** were logged every **10 steps**.
   - At the end of each epoch, average loss, training time, and GPU usage were saved.
   - Regularly, **checkpoints** were saved, including the model, optimizer, and scheduler. So training could be resumed if interrupted.

Diffusion Model: Validation and Testing

Validation

1. We loaded the **final model** (from the 20th checkpoint), restoring all parts: **VAE, UNet, tokenizer**, and **text encoder**.
2. The **UNet** was set to **evaluation mode** (no training updates), and folders were prepared to save outputs.
3. Each batch of validation images was:
   - Encoded into latent space
   - Reconstructed back to image form
   - **Denormalized** so we could visually compare them to the originals.
4. We measured reconstruction quality using:
   - **MSE** (pixel error)
   - **PSNR** (image clarity)
   - **SSIM** (visual similarity)
5. We also **generated new images from text prompts**, saved them, and measured their quality.
6. For both reconstructions and new images, we computed **FID** scores using the cleanfid library to compare them with real images.
7. All images, results, and charts were saved to check how well the model performed.

Testing

1. The **best model checkpoint** was loaded, and the full pipeline was set to evaluation mode.
2. For each test batch, we saved:
   - The original images
   - Their reconstructions
   - New images generated from prompts
3. We calculated the same quality metrics: **MSE, PSNR, SSIM, and FID**.
4. At the end, we averaged all results to get a final summary of how well the model worked on **unseen test data**.

Diffusion Model: Evaluation and Final Metrics

Training Results

- The model trained for **20 epochs**, each with **428 batches**.
- Training loss **steadily decreased**, starting at around **0.15** and ending at **0.0417**, with a final average of **0.1322**.
- The learning rate was fixed at **2e-5**, and peak GPU memory usage was about **39.5 GB**.
- Each epoch took ~3 minutes; full training lasted about **1 hour**.
- Loss curves showed smooth and stable improvement.

Validation Results

- Used the **final model (epoch 20)** to evaluate **1,467 validation images**.
- Reconstruction loss (MSE) was very low: **0.0011**.
- Image quality scores were high: **PSNR 36.29**, **SSIM 0.9561** (very similar to original faces).
- FID scores:
  - **30.85** for reconstructions (VAE)
  - **104.35** for new generations (DDPM)
- Results show the model **reconstructs faces very accurately**, and **generates new faces decently**, with some room for realism improvement.

Test Results

- On **unseen test images** (1,467 samples), the model performed consistently
- FID scores were also close to validation, Confirming the model **generalizes well** and maintains strong performance on new data.
  - **32.67** for reconstructions (VAE)
  - **102.86** for new generations (DDPM)

Category	**Metric**	**Value**
Training	MSE	0.1322
	Learning Rate	2e-05
Validation	MSE	0.0011
	PSNR	36.29
	SSIM	0.96
	FID (VAE)	30.85
	FID (DDPM)	104.35
Testing	MSE	0.0010
	PSNR	36.29
	SSIM	0.96
	FID (VAE)	32.67
	FID (DDPM)	102.86

Diffusion Model: How Our Generated Images Improved: From Noise to Real Faces

From Noise to Faces with Prompt Guidance

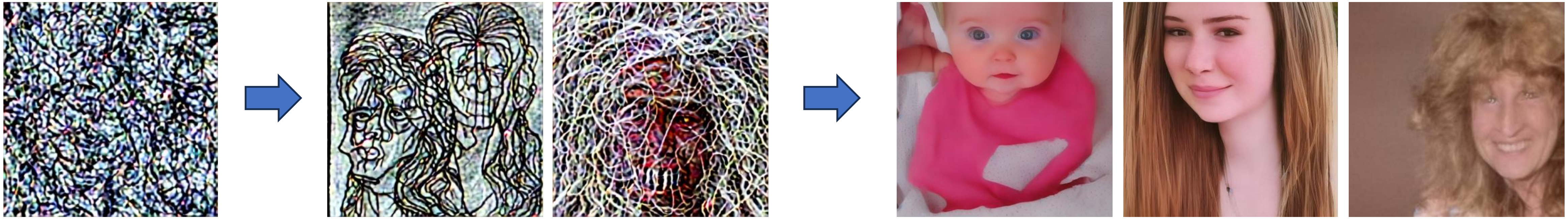
- At first, the model generated only **random noise** (very high FID ~500) because no text prompts were given.
- Once we added **natural language prompts** using the **CLIP text encoder**, the model started generating **face-like features**.
- Prompts helped the model understand **age, gender, and ethnicity**, turning static noise into **recognizable human faces**.

Key Technical Fixes for Major Quality Gains

- Used the **original DDPMScheduler**, noise scheduling algorithm originally used in the Denoising Diffusion Probabilistic Model (DDPM).
- Applied the correct **VAE scaling factor** (0.18215) to match Stable Diffusion's expected range.
- **Fine-tuned only the UNet**, while freezing the text encoder -> its weights were not changed during training.
- Increased the **guidance scale** from 1.0 to 7.5 so the output would better match the prompts.
- Used **prompt dropout** during training to help the model work with or without prompts.

Final Results: Realistic, High-Quality Faces

- With all these improvements, the model produced **clear and realistic faces** based on simple text inputs.
- The **FID score dropped** from ~500 to **about 102**, showing big quality gains.
- The final pipeline was **robust and consistent**, accurately generating faces that matched the given descriptions.



Diffusion Model: Generating Samples and Analysis

Sample Generation

- After fine-tuning, we used the model to **generate face images** from **text prompts** describing age, gender, and race.
- Each prompt produced a **unique face**, and all outputs were **saved for analysis**.

Why Multiple Faces Sometimes Appear

- When generating in **batches**, tools often **combine images into a grid**, making it look like one image with multiple faces.
- If the prompt is **not specific**, the model might generate **group photos**, especially if trained on data with group scenes.
- Using **prompt dropout** during training teaches the model to handle both **solo and group images**.

How to Get Only One Face per Image

- **Generate one image at a time**, not in batches.
- Use **clear prompts** like: *"solo portrait"* or *"one person"*.
- **Increase the guidance scale** to force the model to follow the prompt more strictly.

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

Checking CP Age Estimator on Generated Images By Diffusion Model

How We Evaluated

- We tested our **Conformal Prediction (CP) age and gender estimator** on faces generated by the diffusion model.
- For each image, we compared the **prompted age and gender** with the **model's prediction and confidence interval**.

What We Found

- For **very young or very old** faces, the CP model gave **wide confidence intervals** (up to ±46 years), showing high uncertainty.
- For **adults (20–45 years)**, intervals were **much narrower** (around ±14 to ±22 years), and predictions were usually **close to the prompt**.
- **Gender predictions** were correct about **76% of the time**, with confidence scores indicating the model's certainty.
- Borderline cases had lower confidence, as expected.

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

Checking CP Age Estimator on Generated Images By Diffusion Model

What the table shows

- The **CP age estimator** predicts both the most likely age and a 90% confidence interval for each generated face, showing how certain (or uncertain) the model is about each age prediction.
- The **Half-Width (±) of the confidence interval** is much smaller for adult faces (ages 20–45), meaning the model is more confident, but is much larger for very young or very old faces, where the model is less certain.
- This means the model's predictions are most reliable for adults, while for children and elderly faces, the wide intervals highlight greater uncertainty and caution in interpreting those results.

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

Conclusions & Future Work

Conclusions

- We built a full pipeline combining a **fine-tuned diffusion model** for face generation and a **conformal prediction (CP) model** for age and gender estimation.
- The diffusion model generated **realistic, high-quality faces** that matched age, gender, and race prompts.
- The CP model provided **accurate age predictions** with **well-calibrated confidence intervals**, and **robust gender classification**.
- The system worked best for **adult faces**, with **narrow intervals and high accuracy**.
- **Uncertainty increased** for **very young or elderly** faces in both real and synthetic data.
- Our training and evaluation methods were validated by **strong metrics and visual results**, but also showed areas needing improvement (e.g., better data balance, rare case handling).
- The work shows that combining advanced generative models like Diffusion with uncertainty-aware predictors such as conformal prediction creates trustworthy, interpretable AI pipelines, where both realism and reliability are needed.

Future Work

- Use a **face-specific pre-trained model** (instead of MobileNetV2) in the CP pipeline to improve feature understanding and prediction accuracy.
- Enforce the Diffusion model to generate images containing only one person per output, by improving prompt design and sampling strategy.

References & Tools

References

- **UTKFace Dataset** – Large-scale dataset with age, gender, and race annotations.
  https://www.kaggle.com/datasets/jangedoo/utkface-new
- **Stable Diffusion v1.5** – Pre-trained diffusion model used for image generation.
  https://huggingface.co/runwayml/stable-diffusion-v1-5
- **Denoising Diffusion Probabilistic Models** – Original paper introducing the diffusion framework.
  Ho et al., NeurIPS 2020
  https://arxiv.org/abs/2006.11239
- **Label Distribution Age Encoding (LDAE)** – Robust method for age estimation in images.
  Gao et al., Neurocomputing, 2017
- **Conformal Prediction** – Statistical technique for generating reliable confidence intervals.
  https://www.stat.uchicago.edu/~rigollet/PDF/teaching/conformal.pdf

Key Libraries and Tools

- **PyTorch:** Deep learning framework used for model development
- **TensorFlow / Keras:** Used for data augmentation and model experimentation
- **Diffusers:** HuggingFace library for diffusion model training and inference
- **Transformers:** HuggingFace library for text encoding and CLIP usage
- **NumPy**, **pandas:** Data processing and numerical operations
- **scikit-learn:** Metrics and validation tools
- **matplotlib**, **seaborn:** Plotting and visualization
- **Tqdm:** Progress bars for training and data loading tasks