# Predicting Eye Gaze

Jabed Miah, Gurpreet Singh, Prince Rana, Joceline A. Borja
*Dept. of Computer and Info. Science, CISC-3280 Machine Learning for Neural and Biological Data*
*Fordham University*
Bronx, New York, USA
gsingh60@fordham.edu, jmiah5@fordham.edu, prana@fordham.edu, jaa18@fordham.edu

***Abstract*—This study presents a representation of a multi-model approach to 3D eye gaze estimation using a dataset of 50,000 eye images. We used Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Vision Transformers (ViTs), and Random Forests (RFs). Our goal was to predict a 3D gaze direction vector of synthetic eye images given only eye region and supporting head pose information as inputs. We developed each model using the tensorflow and scikit learn libraries in python. Through rigorous experimentation of varying parameters, we compared each model to eachother. The CNN model outperformed all of the other models, achieving a Mean Squared Error (MSE) of 0.0001 and an $R^2$ score of 0.9980, followed by ViT and RNN. The RF model, while useful, showed the lowest predictive accuracy. Analysis revealed that closed eyes and visual obstructions (like camera clipping) significantly degraded model performance. Despite these challenges, the study successfully demonstrated the feasibility and robustness of deep learning-based gaze estimation. This work was aimed to lay a foundation for future studies in gaze-based human-computer interaction, medical diagnostics, and real-time applications in virtual environments.**

***Index Terms*—Random Forest, Neural Networks, Convolutional Neural Networks, Recurrent Neural Networks, Prediction, Supervised Learning, Ensembles, Sci-kit-learn, TensorFlow, Eye Gaze, Eyes**

## I. Introduction

Human gaze direction plays a crucial role across a wide range of applications, including medical diagnostics, assistive technologies, driving assistance systems, virtual reality, and human-computer interaction. Gaze tracking has become an essential tool in medical research and patient care, particularly for individuals with conditions such as Amyotrophic Lateral Sclerosis (ALS), Autism Spectrum Disorder (ASD), Attention-Deficit/Hyperactivity Disorder (ADHD), Parkinson's disease, and Alzheimer's disease. In these contexts, eye gaze analysis supports clinical reporting, therapy, rehabilitation, and even communication enhancement for patients with limited motor abilities.

In recent years, machine learning-based gaze estimation models have gained traction for their potential to predict where a person is looking based on visual features such as eye appearance and head orientation. Accurate gaze prediction can improve the accessibility, responsiveness, and interactivity of digital systems we use in everyday life, from smarter interfaces to more immersive virtual experiences.

This research focuses on building and evaluating machine learning models capable of predicting 3D gaze vectors from images of the human eye region. We utilize a publicly available dataset from Kaggle containing over 50,000 grayscale eye images, along with corresponding head pose information and gaze vectors. By framing gaze prediction as a regression problem, we aim to develop models that estimate the continuous direction of a person's gaze in 3D space.

To explore the performance of various modeling strategies, we implement and compare multiple approaches, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), Vision Transformers (ViTs), and a traditional ensemble model: Random Forest regressors. Each model is trained and evaluated on the same dataset under controlled conditions, with performance assessed using standard metrics such as Mean Squared Error (MSE) and the coefficient of determination ($R^2$ score).

## II. Background

As technology continues to evolve at an unprecedented pace, 2025 is poised to be a landmark year. What many are calling a "technical year", marked by increased integration of smart systems and artificial intelligence in everyday life. One such area gaining significant momentum is eye-gaze tracking, which has proven to be a powerful tool in fields ranging from medical diagnostics to human-computer interaction. [1] Its relevance is only growing, as new applications emerge in automotive safety, accessibility tools, augmented and virtual reality, and neurology.

Eye-gaze tracking involves determining where a person is looking, typically by analyzing the position and movement of their eyes in relation to the head and screen. The utility of this technology spans a wide spectrum. For instance, it supports communication for individuals with motor impairments, enables targeted advertising and adaptive interfaces in digital environments, and helps researchers understand visual attention and cognitive processes. [2] In medicine, eye tracking is used in diagnostic systems for neurodegenerative diseases, developmental disorders, and post-stroke rehabilitation. In automotive applications, it enhances driver monitoring systems for fatigue detection and distraction prevention. [3]

From a technical perspective, eye-gaze estimation can be broadly categorized into two approaches: hardware-based and software-based. Traditional systems rely on infrared cameras and dedicated sensors to track eye position with high precision. However, these systems are often expensive and limited in accessibility. In contrast, software-based solutions aim to achieve similar outcomes using widely available hardware like webcams and mobile device cameras, relying on computer vision and machine learning algorithms to interpret gaze direction from image data.

Machine learning approaches to gaze estimation typically frame the task as a regression problem, where the goal is to predict a continuous 2D or 3D gaze vector. These models learn from large datasets of labeled eye images and corresponding gaze directions, extracting patterns in pupil position, eyelid shape, iris texture, and head pose. Features such as eye asymmetry, lighting conditions, and individual anatomical variation introduce complexity to the problem, requiring robust models that can generalize across subjects and environments.

In this study, we use a rich, publicly available dataset that includes tens of thousands of grayscale eye images, each annotated with head orientation and a ground truth 3D gaze vector. By leveraging this dataset, we aim to explore how different machine learning models, ranging from traditional ensemble methods to modern deep learning architectures, can be trained to predict where a person is looking, based solely on a static image of their eye region. The ability to accurately infer gaze direction from such minimal input has the potential to revolutionize interactive systems, making them more intuitive, responsive, and accessible across diverse populations. [4]

## III. Experimental Methodology

This section outlines the experimental process followed to build our machine learning models for gaze prediction from grayscale eye-region images. Our objective was to train models that can accurately estimate a person's gaze vector (the direction they are looking at represented in a 3d space using vectors with magnitudes) using only visual and geometric input features derived from a publicly available dataset. The implementation utilizes core Python libraries such as TensorFlow [5] for deep learning architectures, Scikit-learn [6] for ensemble methods and Matplotlib [7] for evaluation and visualization. We explored four types of models in this study, each requiring specific data formatting or architecture-aware preprocessing:

- Convolutional Neural Networks (CNNs): to learn spatial features in the eye region. [8]
- Recurrent Neural Networks (RNNs): to test sequential modeling of image rows as time steps. [9]
- Vision Transformers (ViTs): to apply patch-based attention on fixed-size image grids. [10]
- Random Forest Regressors: to serve as a traditional machine learning baseline using flat features. [11]

### *A. Dataset Description*

The dataset used in this study is the Eye Gaze Dataset [4], hosted on Kaggle, which provides high-resolution grayscale eye images along with associated metadata. The dataset includes over 50,000 images, each paired with annotations describing the head pose and gaze direction of the subject. For this study, we only used the dataset's synthetic images, which were generated in the 3D game engine called Unity. We were unable to use the real eye images for our study because we did not have a way to accurately measure the eye's gaze vectors. The data set includes:

- Image folders: MPIIGaze, normalized, and original images.
- Metadata files: gaze.csv, gaze.h5, real_gaze.h5, and gaze.json

The primary label used for prediction was the look_vec field found in gaze.csv. It contains the 3 dimensions, take x, y and z for example. This vector served as the target output for all of our models. The dataset also included information related to the head positioning, iris, lighting details, and much more, but this study is primary focused solely on predicting the gaze vector based on image input. We did not incorporate those additional features.

### *B. Data Preprocessing*

To support each of our models, we handled the preprocessing with several standards but also slightly tailored it for each model specifically.

1) All of the models were given normalized data. The grayscale images were scaled to pixel values between [0,1]. The raw image shape was (35,55), which worked directly for CNNs and RNNs.
2) ViT were padded to (64,64,1) to meet the input shape requirements of our ViT encoder.
3) We flattened the images to a 1 dimensional vector to feed into the random forest. Since the images were 35x55, we get 1925 data values.
4) We also split the data multiple ways for the models. The first split was 70/15/15. 70% training set, 15% test set and 15% validation set. This was done in the beginning for 5,000 instances of the images instead of entire dataset so we can fine tune our models to see the best parameters. For the entire dataset, we split the dataset into 66% training, 17% test, and 17% validation.

### *C. Model Compatibility with Preprocessing*

Each model architecture benefited from specific preprocessing steps.

- CNNs: Used normalized 2D eye images to capture texture patterns across spatially close pixels. [8]
- ViTs: Required square input sizes; thus, padding to 64x64 was essential for consistent patch division. Patch division refers to the process of dividing an input image into fixed size, non-overlapping patches. (i.e. 16x16 pixels). Each patch is then flattened and treated as a token, similar to words in a sequence for NLP tasks. Then the patches are fed into the transformer for model processing.
- RNNs: Treated each image row as a timestep, so the 2D structure was interpreted as a sequence. Sequences are ordered sets of data points, along some dimension where the order has importance. Each data point in the sequence

is called a time step/element and the RNN was designed to handle our images. [9]

- Random Forests: Operated on flattened vectors to compare performance against deep learning models.

## D. Evaluation Metrics

Performance was primarily assessed using Mean Squared Error (MSE) on the gaze vector predictions. In some experiments, the $R^2$ (coefficient of determination) was also computed to quantify how well the model explained variance in the data.

Mean Squared Error (MSE) is a common metric used to evaluate how well a machine learning model performs in regression tasks. The goal is to predict continuous values, such as a direction vector in our gaze estimation task. Mathematically, MSE is calculated as:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{1}$$

$$\begin{aligned} y_i &: \text{true value} \\ \hat{y}_i &: \text{predicted value} \\ n &: \text{total number of samples} \end{aligned}$$

The result is the average of the squared differences between the actual and predicted values. The squaring ensures that large errors are penalized more than small ones. A lower MSE indicates that the model is making predictions that are closer to the true values. If the MSE is 0, that means the model predicted every value exactly right. R2 or the coefficient of determination, is another metric used by regression models. It measures how much of the variance in the target variable is explained by the model. It indicates how well the model explains the variance in the data. An R2 of 1 indicates perfect prediction. R2 Formula:

$$R^2 = 1 - \frac{RSS}{TSS} \tag{2}$$

$$\begin{aligned} R^2 &: \text{coefficient of determination} \\ RSS &: \text{sum of squares of residuals} \\ TSS &: \text{total sum of squares} \end{aligned}$$

However, a high $R^2$ doesn't always mean the model is good. If the model memorizes the training data instead of generalizing to new examples, it may achieve a high $R^2$ on training data but perform poorly on new data. This is called overfitting. So, while R2 is useful, it must be interpreted carefully and in combination with other metrics like MSE.

## E. Visualization & Error Analysis

For deeper insight into model behavior, we performed the following:

- 2D Gaze Vector Plotting: Using matplotlib's plotting module, we projected true and predicted gaze vectors from the eye center into 2D space. This helped us understand the model's output.
- Best/Worst Predictions: We ranked test samples by vector error magnitude and visualized the best and worst predictions in a grid. This revealed that the worst errors often coincided with low contrast or occluded eye regions.

## IV. Models

### A. Convolutional Neural Network (CNN)

Convolutional Neural Networks (CNNs) are deep learning models particularly well-suited for tasks involving spatially structured data like images. In the context of gaze estimation, CNNs are effective at learning local patterns in eye images such as contours, eyelid shapes, and pupil positioning, which are all crucial for predicting gaze direction. We designed a CNN model tailored to 64x64 grayscale eye images. The goal was to regress a 3D gaze vector [x, y, z] representing the gaze direction. Below is a detailed architecture of the network and rationale behind the chosen layers:

TABLE I
CNN Architecture

| **Layer** | **Description** |
|---|---|
| Input | 64x64 grayscale image (shape: 64x64x1) |
| Conv2D | 32 filters, kernel size (3x3), activation='ReLU'. Captures low-level features like edges. |
| MaxPooling2D | Pool size (2x2). Reduces spatial dimensions and computation. |
| Conv2D | 64 filters, kernel size (3x3), activation='ReLU'. Learns more abstract features |
| MaxPooling2D | pool size (2x2). Further downsamples feature maps. |
| Flatten | Converts 2D feature maps to a 1D vector for dense layers |
| Dense | 128 units, activation='ReLU'. Fully connected layer to learn complex representations. |
| Dense | 64 units, activation='ReLU'. Intermediate layer to further refine features. |
| Output (Dense) | 3 units, activation='linear'. Predicts continuous gaze vector components. |

Below is a list of training details for our CNN.

- Loss Function: Mean Squared Error (MSE). Suitable for continuous regression problems.
- Optimizer: Adam optimizer. Chosen for its adaptive learning rate and stability during training.
- Metrics: MSE and $R^2$ (coefficient of determination) were tracked to assess both absolute error and relative explanatory power.

We have taken great inspiration from Professor Dent's ipynb on fashion MNIST. [12] CNNs use layers that apply filters (kernels) over the image to detect these features. The basic structure involves convolution layers to detect patterns like corners or edges, pooling layers to reduce the image size while keeping important information and fully connected layers to combine all the learned features to make the final prediction. We used a simple CNN model with two convolution layers followed by pooling layers, a dropout layer to prevent overfitting, and a final dense layer to output our gaze vector.

CNNs being inherently good at spatial hierarchies, learning patterns from pixels to abstract concepts through convolution and pooling, helps with gaze estimation. Predicting the gaze vector requires precise recognition of subtle eye cues (like iris position and eyelid curvature), CNNs offer a strong inductive bias over other models by constraining the model to local pixel connectivity and translation invariance. The CNN provides a strong performance in terms of both speed and predictive accuracy, which makes it an effective candidate especially for possible deployment in lightweight or real-time applications.

## B. *Vision Transformer (ViT)*

Vision Transformers (ViTs) represent a fundamental shift from traditional convolutional approaches to image understanding. Inspired by the success of Transformers in Natural Language Processing (NLP), ViTs treat images not as continuous spatial arrays but as sequences of image patches, analogous to words in a sentence. This reimagining enables the model to learn long-range dependencies and global relationships that CNNs may miss due to their local receptive fields.

Comparing to traditional CNNs, the CNN operates on the premise of local connectivity, they learn spatial hierarchies by convolving filters across local regions. ViTs, on the other hand, discard this assumption. Instead, they:

- Divide an image into fixed-size patches (like ”tokens”)
- Flatten and embed each patch into a vector space
- Use positional embeddings to preserve the order of patches
- Pass these through a stack of transformer encoder layers using Multi-Head Self-Attention (MHSA) to capture relationships between patches globally
- And finally pool the information for downstream tasks like regression or classification

This method allows ViTs to learn contextual dependencies across the entire image from the outset, potentially improving performance on tasks like gaze estimation where subtle, spatially-distributed cues (e.g., eyelid position, iris shape) can inform prediction.

Unlike prebuilt implementations from high-level libraries like Hugging Face, we constructed our ViT from scratch in TensorFlow/Keras to fully control architecture and deepen our understanding.

### *1) Input and Preprocessing:*

- Input shape: 64x64 grayscale images (1 channel)
- Patch size: 8x8
- Number of patches: (64/8)2̂=64
- Flattened patch dimension: 8x8x1=64

### *2) Patching and Embedding:*

1) Patch Extraction: Each image was divided into 64 non-overlapping patches. Patches were flattened into 64 dimensional vectors.
2) Patch encoding: Each flattened patch was projected into a 65 dimensional embedding vector via a dense layer. Learnable positional embeddings were added to retain spatial order, resulting in a sequence of 64 patch embeddings of size 64.

*3) Transformer Encoder Architecture:* Each encoder block followed this pattern, repeated 4 times:

TABLE II
VIT TRANSFORMER ENCODER ARCHITECTURE

| **Component** | **Description** |
|---|---|
| Layer Normalization | Normalizes inputs to stabilize training. |
| Multi-Head Self-Attention (MHSA) | 4 heads; each computes its own Q, K, V matrices, attends to different aspects (e.g., edges, spatial context). Attention is computed as: $\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$ |
| Skip Connection (1) | Adds original input to the attention output to preserve information. |
| Layer Normalization (2) | Further normalization before MLP. |
| Feedforward MLP | Dense $\rightarrow$ Activation $\rightarrow$ Dropout $\rightarrow$ Dense. Applies non-linearity to each embedding. |
| Skip Connection (2) | Adds MLP output back to prior layer to stabilize gradients. |

These stacked layers allow the model to iteratively refine attention across patches, capturing complex global dependencies.

### *4) Output and Regression Head:*

- After encoding, the sequence of patches was flattened into a single vector.
- This vector was passed into an MLP head with fully connected layers for our gaze direction regression.
- The final output layer had 3 neurons with linear activation, corresponding to the predicted 3D gaze vector components [x,y,z].

*5) Training Configuration:* The training configuration uses MSE loss function, adam optimizer, and track the performance with MSE and $R^2$.

TABLE III
VIT ARCHITECTURE

| **Parameter** | **Description** |
|---|---|
| Patch Size | 8x8 |
| Number of Patches | 64 |
| Embedding Dimension | 64 |
| Transformer Layers | 4 |
| FAttention Heads | 4 |
| MLP Head Units | 64,32 |
| Final Output | 3 (linear units) |

Gaze direction can be influenced by subtle variations across the entire eye region: pupil location, eyelid openness, and even indirect contextual features. ViTs are uniquely positioned to capture both local and global cues without the inductive biases of convolutions. This enables potentially greater generalization, particularly across diverse lighting, skin tones, and eye shapes.

### C. Recurrent Neural Network (RNN)

Recurrent Neural Networks (RNNs) are designed to model sequential dependencies in data. While traditionally used for natural language processing or time-series forecasting, their architecture makes them a viable tool for image-derived sequences, especially when features are unrolled spatially or over time. In our case, we prepared the data as a sequence of 55 time steps with 35 features per step. These could represent derived, flattened image patches, eye movement signals, or temporally structured visual data. Each "step" acts like a frame in a sequence, allowing the RNN to model temporal or structural patterns across the input.

TABLE IV
RNN LAYERS

| Layer | Details |
|---|---|
| GRU Layer 1 | 64 units, returns full sequence (one output per timestep) |
| Dropout | Applied after GRU1 to reduce overfitting |
| GRU Layer 2 | 32 units, returns final hidden state only |
| Transformer Layers | 4 |
| Dropout | Again applied to reduce overfitting |
| Dense Layer | 16 units with ReLU refines representation |
| Output Layer | 3 units, linear activation outputs 3D gaze vector |

*1) RNN Architecture:* Why GRU? Gated Recurrent Units (GRUs) are a more efficient alternative to LSTMs. They use fewer gates while still learning long-term dependencies, which reduces computational cost without significant accuracy loss. The training configuration uses MSE loss function, adam optimizer, and track the performance with MSE and $R^2$.

TABLE V
RNN PARAMS

| Parameter | Value |
|---|---|
| Input Shape | (55,35) |
| GRU Units | $64 \rightarrow 32$ |
| Dense Units | 16 |
| Output Units | 3 |
| Dropout Rate | Default |

While not spatial in nature like CNNs or ViTs, RNNs allow us to treat preprocessed image features or patch embeddings as sequences, learning how different "patch states" influence each other over steps. This offers an alternate view of the gaze estimation task, one where structural dependencies are modeled as sequential patterns.

### D. Random Forest (RF)

Random Forest is an ensemble learning algorithm based on decision trees. Rather than relying on a single tree, it builds a collection (or forest) of decision trees and averages their predictions. This reduces overfitting and enhances the model's ability to generalize on unseen data. For a regression task like ours, predicting a continuous 3D gaze vector, this method is well-suited due to its robustness, interpretability, and capability to model complex nonlinear relationships.

Our task required predicting precise directional vectors based on visual information. Random Forest's ensemble approach helps smooth out individual tree errors, especially valuable when dealing with moderately noisy data and high-dimensional inputs like flattened image pixels.

Unlike our previous models, random forest underwent several changes. The train test split was 80% test and 20% test. There was no validation set. Preprocessing was also slightly different. The feature head_pose was also used as an input.

TABLE VI
RANDOM FOREST PARAMETERS

| Parameter | Value |
|---|---|
| n_estimators(# of trees) | 100 |
| max_depth | None (nodes expand until pure) |
| random_states | Default(42) |
| n_jobs | -1 (utilize all CPU cores) |

These settings were chosen to balance model accuracy and training efficiency. Notably, max_depth=None allows each tree to grow fully, enabling more complex decision boundaries at the risk of overfitting, which is countered by averaging across 100 trees.

TABLE VII
RANDOM FOREST

| Element | Description |
|---|---|
| Feature Dimensionality | 4096 (image) + 3 (head pose) |
| Output Shape | 3 (X, Y, Z gaze direction) |
| Model Type | Non-parametric, ensemble method |
| Biggest Pro to RF | Handles non-linearity, interpretable, robust to outliers |
| Worst Con to RF | High memory usage, slower inference |
| Library | scikit-learn |

## V. RESULTS

Each model was evaluated on the test set using Mean Squared Error (MSE) and the $R^2$ Score, which measures the proportion of variance in the target explained by the model. Lower MSE and higher $R^2$ values indicate better performance. To dip deeper, the top 5 angle errors were recorded of best and worst types. They were also visualized using matplotlib. We included two of the best and two of the worst for the CNN, ViT and RNN.

### A. CNN Results

The CNN achieved the best overall performance, with the lowest error and highest explained variance. Its ability to extract spatial features from eye images made it particularly well-suited for this task. The model's balance between depth and regularization contributed to its generalization power.

TABLE VIII
CNN Results

| Metric | Value |
|---|---|
| MSE | 0.0001 |
| $R^2$ | 0.9980 |

Digging deeper, We have also observed the angle errors present in the model's predictions in degrees.

*1) Examples of Error:* They have been ranked and plots of examples are present.

TABLE IX
Best Performing CNN

| Ranking | Error Value |
|---|---|
| 1 | 0.00000000 |
| 2 | 0.00351414 |
| 3 | 0.00534144 |
| 4 | 0.01158251 |
| 5 | 0.01636623 |

TABLE X
Worst Performing CNN

| Ranking | Error Value |
|---|---|
| 1 | 18.54289976 |
| 2 | 16.22099673 |
| 3 | 15.14922767 |
| 4 | 9.85527177 |
| 5 | 8.77415607 |

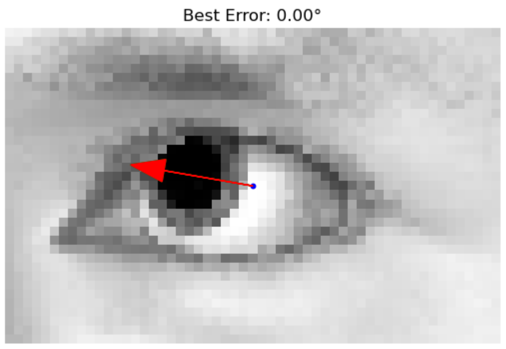


Fig. 1. 2nd Best

## B. ViT Results

The ViT model performed nearly as well as the CNN, demonstrating strong capability in capturing global patch-level relationships through self-attention. Although it had slightly higher error, it validated that transformer-based models can be effective in image regression tasks when trained from scratch.

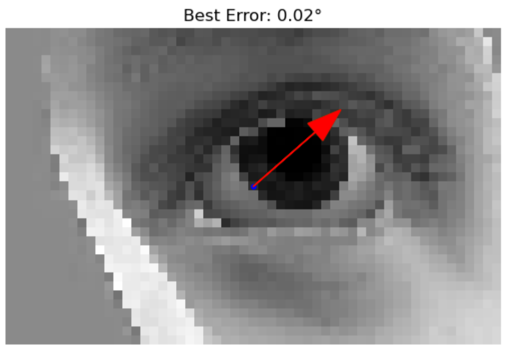


Fig. 2. 5th Best

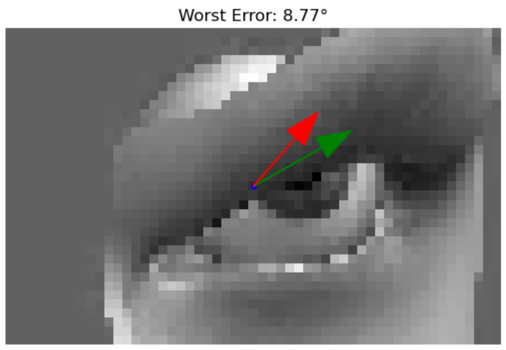


Fig. 3. 5th Worst

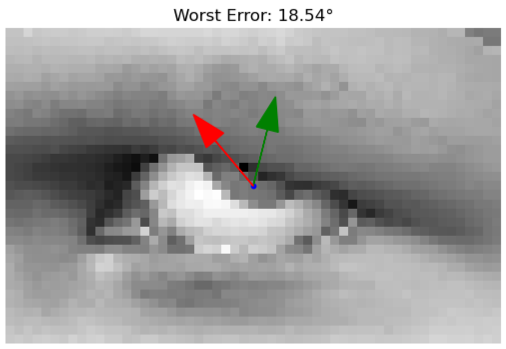


Fig. 4. 1st Worst

TABLE XI
ViT Results

| Metric | Value |
|---|---|
| MSE | 0.0006 |
| $R^2$ | 0.9921 |

*1) Examples of Error:* They have been ranked and plots of examples are present.

TABLE XII
BEST PERFORMING VIT

| Ranking | Error Value |
|---|---|
| 1 | 0.01082814 |
| 2 | 0.01695111 |
| 3 | 0.02563048 |
| 4 | 0.02873991 |
| 5 | 0.036549 |

TABLE XIII
WORST PERFORMING VIT

| Ranking | Error Value |
|---|---|
| 1 | 41.89421944 |
| 2 | 41.29439635 |
| 3 | 34.40028006 |
| 4 | 32.71560518 |
| 5 | 28.92588613 |

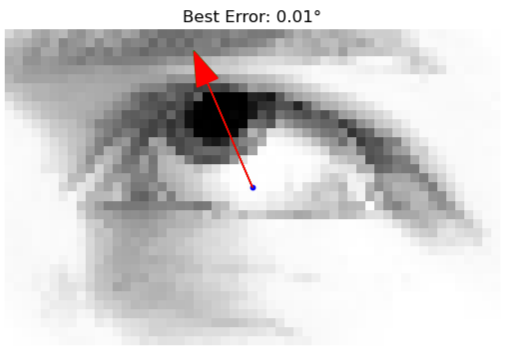


Fig. 5. Best

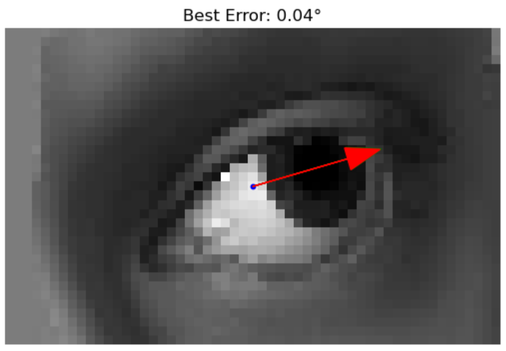


Fig. 6. 5th Best

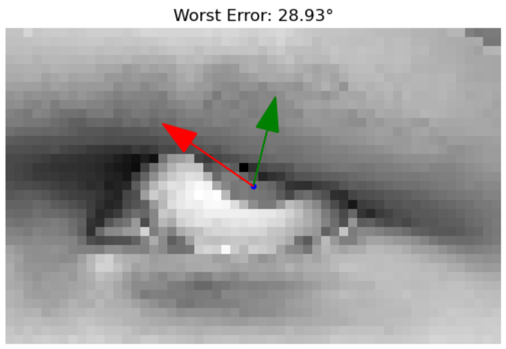


Fig. 7. 5th Worst

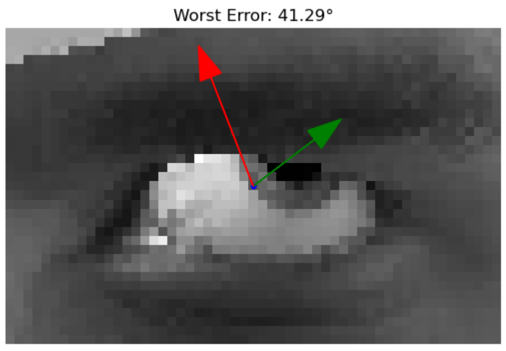


Fig. 8. 2nd Worst

## C. *RNN Results*

TABLE XIV
RNN RESULTS

| Metric | Value |
|---|---|
| MSE | 0.0040 |
| $R^2$ | 0.9427 |

The RNN performed reasonably well, though not on par with CNN or ViT. Since the gaze estimation problem is more spatial than sequential, the RNN's sequential processing did not capture image structure as effectively. However, it still demonstrated value by learning dependencies in the feature sequences.

*1) Examples of Error:* They have been ranked and plots of examples are present.

TABLE XV
BEST PERFORMING VIT

| Ranking | Error Value |
|---|---|
| 1 | 0.01642824 |
| 2 | 0.02429986 |
| 3 | 0.02880105 |
| 4 | 0.03342123 |
| 5 | 0.04628141 |

TABLE XVI
WORST PERFORMING RNN

| Ranking | Error Value |
|---|---|
| 1 | 64.12749293 |
| 2 | 30.77065957 |
| 3 | 28.03501455 |
| 4 | 25.44807122 |
| 5 | 22.42333566 |

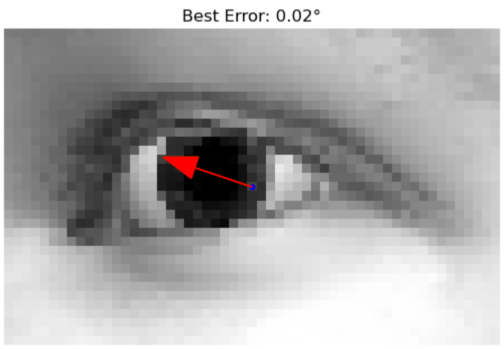


Fig. 9. 2nd Best

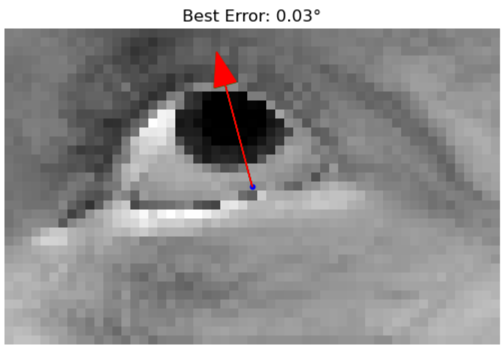


Fig. 10. 4th Best

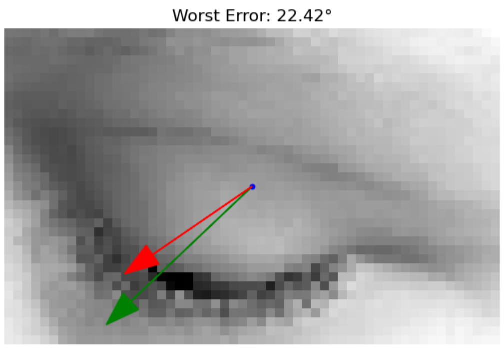


Fig. 11. 5th Worst

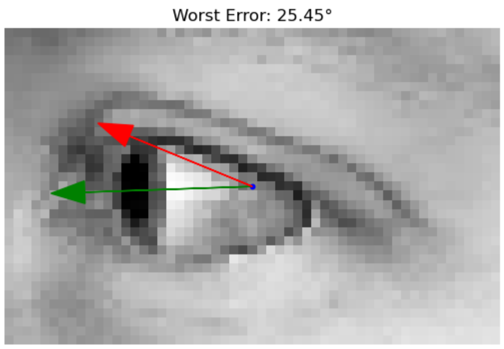


Fig. 12. 4th Worst

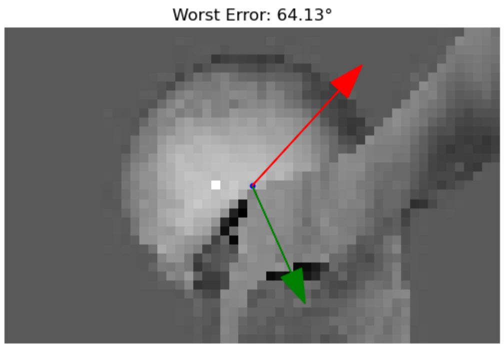


Fig. 13. 1st Worst

### D. RF Results

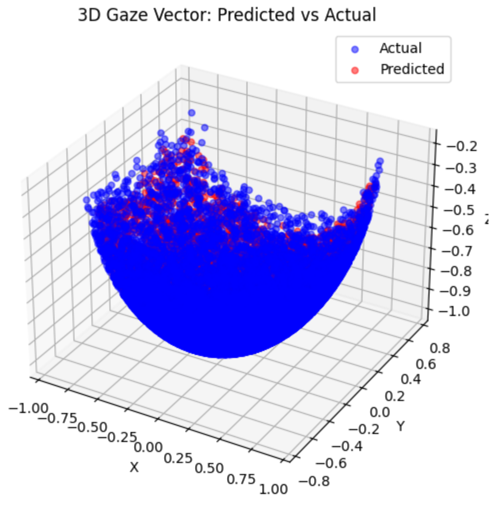
TABLE XVII
RF RESULTS

| Metric | Value |
|---|---|
| MSE | 0.0144 |
| $R^2$ | 0.8417 |

The Random Forest lagged behind the deep learning models. While it handled non-linear relationships between gaze vectors, head pose, and image pixels, its performance was limited by the high dimensionality of the flattened images and the lack of spatial feature extraction.

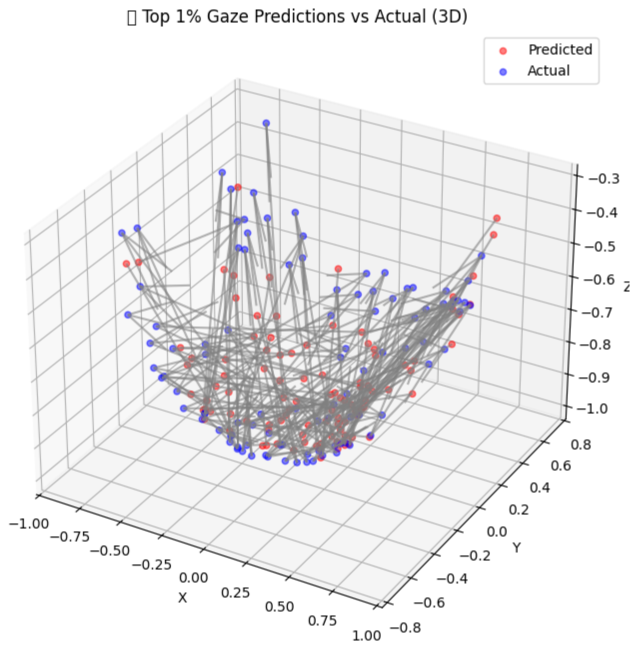
3D Scatter + Quiver Plot (Predicted vs Actual Gaze):

- Red dots: Model's predicted gaze directions.
- Blue dots: Actual true gaze directions.
- Both red and blue can be used to show the distance between the actual gaze point and predicting gaze (error vector).
- Visually shows how far off each prediction was.
- Lets us know if there may be directional bias or systematic errors in the model.



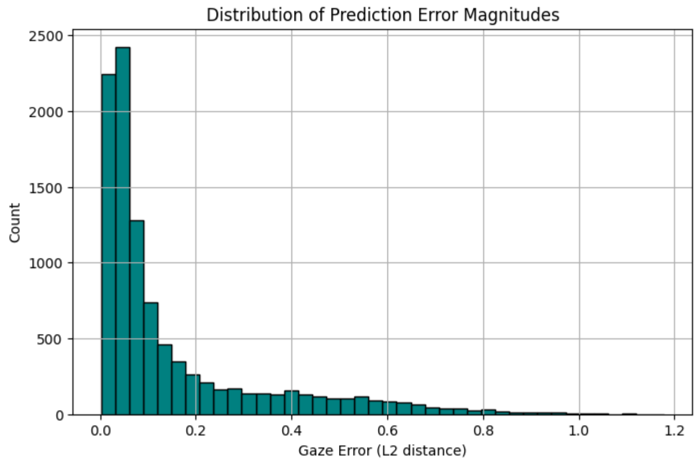
Top 1% Largest Error Vectors (Focused Quiver Plot)

- Focused only on the worst 1% of the predictions (biggest mistakes).
- Plotted error vectors for the largest mistakes.
- Helps zoom in on where and how the model struggled most.
- Useful for future model improvements.
- The grey line shows how far off the predicted value is from target value.



Error Histogram:

- X-axis: Size of prediction error (how wrong the model was).
- Y-axis: Number of samples (how often that error size happened).
- Further helps with identifying how far off the prediction was from the actual values.



## VI. DISCUSSION

The results of our experiments demonstrate the clear advantage of deep learning architectures, particularly Convolutional Neural Networks (CNNs) and Vision Transformers (ViTs), in accurately predicting 3D gaze vectors from grayscale eye images. The CNN model yielded the lowest Mean Squared Error (MSE) and highest $R^2$ score, outperforming all other models tested. Its superior performance can be attributed to its well-established ability to extract spatial hierarchies of features from image data that were not inherently given, particularly local patterns such as iris location, eyelid curvature, and pupil size. These localized features are critical for gaze prediction and are effectively learned through convolutional operations.

The Vision Transformer, while slightly trailing the CNN, also produced strong results, reinforcing the model's ability to capture global spatial dependencies across the image through multi-head self-attention mechanisms. Despite being trained from scratch and on a modest grayscale dataset without pre-training, the ViT exhibited strong generalization. This is consistent with findings in recent literature showing that ViTs, while data-hungry, can match or exceed CNNs in vision tasks given appropriate architectural tuning and input structure [**?**] [13].

In contrast, the Recurrent Neural Network (RNN) and Random Forest (RF) models performed significantly worse. The RNN, although useful for capturing sequential or temporal dependencies, struggled to model spatial relationships in static images. Since image structure is two-dimensional and not inherently sequential, treating image rows as sequences may have not enabled the model to fully learn meaningful gaze patterns as effectively as spatial models. However, the results for RNN are pretty good, it had the worst time dealing with the outliers.

The Random Forest model performed the worst, which is expected given its lack of capacity to learn complex visual features from high-dimensional image inputs. RFs are effective in structured tabular data but are not inherently designed for pixel-level interpretation. Their performance is further hindered when features like pixel values are highly correlated or when spatial structure is ignored, common limitations acknowledged in vision-based random forest studies [14]. The poor performance here could also be partially due to limited hyperparameter tuning and the absence of feature engineering techniques tailored for image data.

*1) Sources of Error:* Despite overall strong results from the CNN and ViT models, we observed several high-error outliers, revealing key limitations in our dataset and modeling setup.

1) Closed or Partially Closed Eyes:
   Across all models, predictions deteriorated when eyes were partially or fully closed. This is likely because the models rely heavily on visible iris and pupil regions for gaze estimation, which become obscured or deformed when eyelids are closed. Without clear landmarks, the models default to weaker contextual features such as eyelid shape or surrounding skin texture, leading to ambiguous outputs.
2) Unrealistic Synthetic Data:
   Several high-error cases involved rendering artifacts, such as when the Unity camera intersected the synthetic head mesh. In such images, the eyeball could be seen through the head or was partially occluded by skin, resulting in visually incoherent data that confused all models. As gaze estimation is highly sensitive to small spatial cues, even minor artifacts can severely mislead predictions.
3) Anatomical and Lighting Variability:
   Models like the RNN and RF, which lack strong spatial inductive biases, showed greater variance in prediction errors. Differences in eyelid shape, skin tone, lighting direction, or iris size introduced inconsistencies that these models were unable to generalize over. CNNs and ViTs, with their structured learning of spatial relationships, were more robust in these situations.

*2) Comparative Performance:* The CNN demonstrated not only the best overall accuracy but also the smallest gap between best-case and worst-case predictions, confirming its robustness across a wide range of eye orientations and conditions. The ViT performed comparably but showed slightly greater variance in edge cases, likely due to its increased sensitivity to spatial perturbations without convolutional locality. The RNN, while adequate in constrained scenarios, had difficulty generalizing in complex visual conditions. Lastly, the Random Forest's performance was hindered by its structural limitations and lack of adaptability to unstructured image data.

These results reinforce a broader finding in the literature: task-aligned inductive biases matter. CNNs, by design, embed priors well-suited for vision tasks, and ViTs, when structured properly, can offer strong alternatives, while models misaligned with the data structure (like RNNs for images) or simplistic models (like RFs) are likely to underperform [**?**] [14].

## VII. Challenges & Future Work

Throughout the course of this project, several challenges emerged which shaped the trajectory of our work and highlighted areas for future improvement. One of the primary difficulties was the sheer size of the dataset. While the large volume of samples (50,000) provided us with rich training data and encouraged experimentation with various model architectures, it also introduced computational constraints that limited how deeply we could tune and optimize each model within our timeframe.

A key technical challenge was visualizing and interpreting the gaze vectors. Initially, we misunderstood how to properly map the look_vec data to 3D space. Our early visualizations incorrectly assumed the gaze direction from the observer's point of view rather than from the person in the image. After experimenting with flipped axes and re-centering the coordinate system, we discovered that the Y component of the look vector needed inversion to align with the subject's perspective. This correction significantly improved the alignment of predicted vectors with actual eye orientation.

Another limitation we encountered was in the dataset itself. Although it contained a mix of synthetic and real eye images, only the synthetic data was usable for training. The real eye images lacked the necessary labeled gaze vectors, restricting their utility to testing and qualitative validation. In future work, we aim to use more real-world datasets that include labeled eye and head pose data, or even collect our own real eye images to train and evaluate model robustness in practical scenarios.

The Random Forest model, while included as a traditional baseline, was not given the same level of architectural optimization or sample-wise analysis as the deep learning models. Its comparatively poor performance may in part reflect suboptimal hyperparameters or an inability to handle the high dimensionality of raw image inputs without preprocessing. Future work could include feature extraction methods—such as eye landmark detection or dimensionality reduction—followed by retraining the RF to assess whether better-tuned classical models could provide competitive performance. Like with our neural models, we could also visualize predictions for best- and worst-case RF outputs to better understand its strengths and limitations.

Beyond gaze estimation, we are interested in expanding the model's capabilities. Potential future tasks include predicting pupil size, iris diameter, or even full head pose estimation. Such enhancements could open the door to meaningful applications such as early detection of neurological conditions (e.g., autism), advanced human-computer interaction in virtual reality environments, or intelligent e-reading systems that respond to user gaze behavior.

Finally, this project did not implement any data augmentation techniques, such as horizontal flipping, brightness variation, or noise injection. Given the grayscale and structured nature of eye images, we believe this is a promising avenue for improving model generalization in future iterations.

Despite these challenges, our results demonstrate that high-performing gaze estimation models can be developed from scratch and trained effectively on synthetic data. These models have potential for broad applicability and serve as a strong foundation for more advanced, real-world systems.

## VIII. Conclusion

This research offers a compelling demonstration of how machine learning can be effectively applied to eye gaze estimation using a large-scale synthetic dataset. Among the four models evaluated, the Convolutional Neural Network consistently delivered the highest predictive performance, validating its suitability for this spatially-aware visual task. While the Vision Transformer and RNN also performed admirably, their sensitivity to noise and sequential complexity introduced greater error in edge cases. The Random Forest model, though capable of capturing non-linear relationships, struggled to match the precision of the deep learning models.

Importantly, this study highlights the power of synthetic data in training effective gaze models while also revealing their limitations, especially in handling real-world anomalies like closed eyes or obstructed pupils. These findings not only confirm the feasibility of gaze estimation with minimal supervision but also open the door for broader applications, such as accessibility technologies, early neurological disorder detection, and adaptive user interfaces.

Looking forward, incorporating real eye data, extending the prediction tasks beyond gaze: iris dilation, and head pose, and augmenting the dataset will be key to making such models more adaptable and applicable in real-world settings. Ultimately, this work builds a solid baseline and points toward a future where machines better understand human intent through subtle visual cues.

## References

[1] Duchowski, A. T. (2007). Eye Tracking Methodology: Theory and Practice. Springer.

[2] Jacob, Robert & Karn, Keith. (2003). Eye Tracking in Human-Computer Interaction and Usability Research: Ready to Deliver the Promises. 10.1016/B978-044451020-4/50031-1.

[3] Harezlak K, Kasprowski P. Application of eye tracking in medicine: A survey, research issues and challenges. Comput Med Imaging Graph. 2018 Apr;65:176-190. doi: 10.1016/j.compmedimag.2017.04.006. Epub 2017 May 30. PMID: 28606763.

[4] 4Quant, K Scott Mader, Eye Gaze Simulated and real datasets of eyes looking in different directions. https://www.kaggle.com/datasets/4quant/eye-gaze

[5] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Zheng, X. (2016). TensorFlow: A system for large-scale machine learning. 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), 265–283. https://www.tensorflow.org https://doi.org/10.48550/arXiv.1605.08695

[6] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine Learning in Python. J. Mach. Learn. Res. 12, null (2/1/2011), 2825–2830. https://scikit-learn.org

[7] J. D. Hunter, "Matplotlib: A 2D Graphics Environment," in Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, May-June 2007, doi: 10.1109/MCSE.2007.55. keywords: Graphics;Interpolation;Equations;Graphical user interfaces;Packaging;Image generation;User interfaces;Operating systems;Computer languages;Programming profession;Python;scripting languages;application development;scientific programming, https://matplotlib.org

[8] Walter Hugo Lopez Pinaya, Sandra Vieira, Rafael Garcia-Dias, Andrea Mechelli, Chapter 10 - Convolutional neural networks, Editor(s): Andrea Mechelli, Sandra Vieira, Machine Learning, Academic Press, 2020, Pages 173-191, ISBN 9780128157398, https://doi.org/10.1016/B978-0-12-815739-8.00010-9. (https://www.sciencedirect.com/science/article/pii/B9780128157398000109)

[9] Alexander Rehmer, Andreas Kroll, On the vanishing and exploding gradient problem in Gated Recurrent Units, IFAC-PapersOnLine, Volume 53, Issue 2, 2020, Pages 1243-1248, ISSN 2405-8963, https://doi.org/10.1016/j.ifacol.2020.12.1342. (https://www.sciencedirect.com/science/article/pii/S2405896320317481)

[10] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2021). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. arXiv:2010.11929 [cs.CV]. https://arxiv.org/abs/2010.11929

[11] Ned Horning, Random Forests: An algorithm for image classification and generation of continuous fields data sets. In Proceedings of the International Conference on Geoinformatics (GeoInfo-Lab). Retrieved from "https://www.geoinfo-lab.org/gisideas10/papers/Random%20Forests%20%20An%20algorithm%20for%20ima

[12] Kyle Dent, CNN for Fashion MNIST https://colab.research.google.com/drive/1Zl2wkCF-OORGN_-lZN2M4Gky9UMN09AKscrollTo=dfmI1eI-jpvL

[13] Touvron, H., Cord, M., Douze, M., et al. (2021). Training data-efficient image transformers & distillation through attention. In ICML. https://doi.org/10.48550/arXiv.2012.12877

[14] J. Shotton et al., "Real-time human pose recognition in parts from single depth images," CVPR 2011, Colorado Springs, CO, USA, 2011, pp. 1297-1304, doi: 10.1109/CVPR.2011.5995316. keywords: Joints;Three dimensional displays;Proposals;Cameras;Shape;Training;Vegetation,