Final Report - Chessboard Image Generation from FEN

1. Abstract

This project addresses the problem of synthetic to real image translation for chessboard rendering, with the goal of transforming synthetic chessboard images into matching realistic looking photographs. We propose a (cGAN) that performs supervised image to image translation from the synthetic renders to the real image domain**.** Results show that the proposed method successfully transforms the synthetic inputs to realistic looking images while maintaining accurate geometric consistency.

2. introduction

Synthetic data is widely used in computer vision to overcome the high cost and limited availability of annotated real world images. In structured domains such as chessboard scenes, synthetic rendering offers full control over board configurations, piece placement, and camera viewpoint, enabling the generation of diverse and perfectly labeled data. However, models trained on purely synthetic images often fail to generalize to real world inputs due to a significant visual domain gap. As a result, synthetic images lack the texture, lighting variations, and noise characteristics present in real photographs. This project focuses on the task of synthetic to real image translation for chessboard rendering, aiming to bridge this visual gap while preserving the exact geometric structure and semantic content of the board.

The primary challenge in synthetic to real translation lies in improving visual realism without altering the underlying scene structure. In the case of chessboards, even minor geometric distortions or piece misplacements can invalidate the generated image for downstream tasks. Furthermore, effective image to image translation models, such as conditional generative adversarial networks, require supervised training on paired synthetic and real images. Obtaining such paired data is difficult and expensive, as real chessboard images are not naturally aligned with their corresponding synthetic representations. Additionally, available chess game data in PGN format provides complete board state information but lacks direct image supervision. The goal of this project is therefore twofold: to design a translation model that produces realistic

chessboard images while strictly preserving board geometry, and to develop a strategy for leveraging unsupervised PGN based data to support supervised model training.

This project makes three main contributions. First, we propose a conditional GAN based image to image translation framework that maps synthetic chessboard renders to the real image domain using single static inputs, without relying on temporal information. Second, we introduce a data generation and labeling pipeline that converts PGN based game states into additional supervised training samples by synthesizing corresponding chessboard images, significantly expanding the effective training dataset. Finally, we conduct a comprehensive experimental evaluation, including qualitative visual comparisons and ablation studies, demonstrating improved visual realism while maintaining accurate board structure and piece placement.

3. Related Work

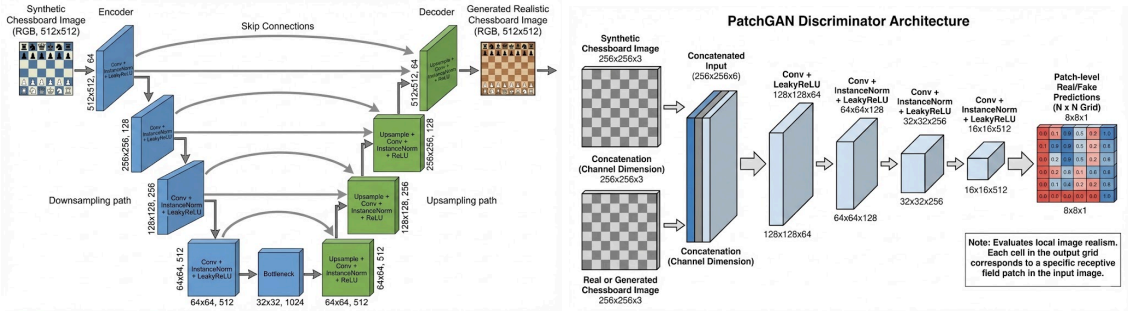
Image to image translation has been widely studied using supervised conditional generative adversarial networks (cGANs). In particular, prior work has shown that cGANs are effective for translating structured synthetic inputs into realistic images when paired supervision is available. This line of work motivates our choice of a supervised cGAN framework for synthetic to real chessboard translation.

Several existing tools support the representation of chess games through symbolic formats such as PGN and FEN. In our pipeline, we use the Fenify utility to convert PGN based game data into explicit board state descriptions. While these tools provide accurate symbolic representations, they do not directly address the challenge of associating board states with visual data.

To obtain supervised image board pairs, we rely on an existing automatic labeling approach that aligns chess video frames with PGN move sequences. This method scores each frame against candidate board states using a deep learning model and applies dynamic programming to determine the most consistent temporal alignment. We use this approach as a data generation mechanism rather than proposing a new alignment algorithm. Additionally, we apply an off the shelf hand detection model based on Google MediaPipe Hands to remove frames with hand occlusions as part of dataset preprocessing.

Unlike prior image to image translation work that assumes naturally paired datasets, our approach integrates these existing tools into a unified pipeline that enables supervised training in a domain where paired visual data is not readily available.

4. Method



4.1 Model Architecture

We adopt a supervised conditional GAN (cGAN) architecture following the pix2pix paradigm. **The generator** is implemented as a U-Net based encoder decoder network with skip connections between corresponding downsampling and upsampling layers. The encoder consists of a sequence of convolutional blocks with stride 2, progressively increasing the number of channels up to 512, followed by a bottleneck layer. The decoder mirrors this structure using nearest neighbor upsampling, convolutional layers, and skip connections to preserve spatial details. The generator outputs an RGB image of the same resolution as the input and applies a tahn activation to constrain pixel values.

The discriminator is implemented as a PatchGAN classifier, which operates on concatenated pairs of input and output images and produces a spatial map of real/fake predictions. The discriminator architecture is configurable in terms of depth and channel width. In our experiments, we use four convolutional layers with a reduced base channel size to balance adversarial strength and training stability. Instance normalization is applied in intermediate layers, and spectral normalization can optionally be enabled, although it was not used in the final configuration.

4.2 Input and Output Representation

The model operates purely on image data. The input to the generator is a synthetic RGB image of a chessboard rendered from a known board position. The output is a realistic

RGB image intended to match the visual appearance of a real chessboard photograph corresponding to the same position. No symbolic information such as FEN or PGN is provided to the model during training or inference. These representations are used only during dataset construction and preprocessing.

All images are resized or warped to a fixed resolution of 512×512 pixels. Pixel values are initially scaled to the range [0, 1] and then mapped to the range [−1, 1] prior to being fed into the network, consistent with the generator's tanh output activation.

4.3 Training Procedure

Training is conducted in a fully supervised manner using paired synthetic–real chessboard images. For each board position, the dataset may contain multiple real images corresponding to a single synthetic render. During training, one real image is sampled at random for each synthetic input, which introduces visual variability while preserving correct supervision. To maximize the effective use of the curated dataset, no explicit train–validation split is employed; instead, the model is trained on the full dataset, and progress is monitored qualitatively.

The generator and discriminator are trained jointly for 200 epochs using the Adam optimizer. The generator learning rate is set to $2\ x\ 10^{-4}$, while the discriminator learning rate is set to $1x10^{-4}$, intentionally weakening the discriminator to promote stable adversarial training. The momentum parameters are fixed to β1=0.5 and β2=0.999. Training proceeds by alternating optimization steps for the discriminator and the generator at each iteration.

To improve stability, an R1 gradient penalty is applied to the discriminator using real samples. The generator is optimized with a composite loss that combines adversarial, reconstruction, and perceptual objectives. Mixed precision training is enabled when supported by the hardware to reduce memory usage and accelerate training.

Model checkpoints are saved periodically throughout training. After each checkpoint, the generator is evaluated on a fixed set of synthetic test images, and the resulting outputs are saved for qualitative inspection. This visual evaluation process allows us to track convergence, assess realism improvements, and detect potential training failures such as mode collapse.

4.4 Loss Functions

The generator is trained using a weighted combination of adversarial, reconstruction, and perceptual loss terms, each serving a distinct role in the image translation process.

The **adversarial loss** encourages realism in the generated images by training the generator to fool the discriminator. This loss promotes realistic textures, high frequency details, and global visual coherence that cannot be achieved through pixel wise supervision alone. Without the adversarial component, outputs tend to appear overly smooth and lack realistic visual characteristics.

The **L1** reconstruction loss enforces structural consistency between the generated image and the corresponding real image. By penalizing absolute pixel wise differences, this loss encourages the preservation of chessboard geometry, including square layout and piece placement. Compared to L2 loss, L1 loss is less sensitive to outliers and helps reduce excessive blurring. In our final configuration, this term is assigned a high weight to ensure accurate board structure.

The **perceptual loss** is computed using feature activations from intermediate layers of a pretrained VGG-16 network. By comparing images in a learned feature space rather than at the pixel level, this loss captures higher level visual similarity such as shapes and textures, improving perceptual quality when exact pixel alignment is not achievable.

The overall generator objective is defined as:

$$L_G = L_{adv} + \lambda_{L1} \cdot L_{L1} + \lambda_{perc} \cdot L_{perc}$$

where the L1 reconstruction loss is weighted by $\lambda_{L1} = 100$ and the perceptual loss by $\lambda_{perc} = 10$. This weighting emphasizes correct structural alignment while allowing the adversarial and perceptual components to refine visual realism.

4.5. Dataset Construction, Preprocessing, and Postprocessing

4.5.1 Dataset Construction from PGN using Fenify and Automatic Labeling and Alignment

To automatically generate frame level ground truth labels for chess video data, we align each video frame with a corresponding board position derived from a PGN game record. This process combines neural board recognition using Fenify-3D with a principled sequence alignment procedure that accounts for temporal continuity and camera orientation.

Fenify-3D Output Representation

For each video frame **t**, Fenify-3D produces a dense probabilistic representation of the chessboard. Rather than predicting a discrete FEN string directly, the model outputs logits for each square independently, resulting in a tensor of size: **64 squares×13 classes**

The 13 classes correspond to the 12 possible chess piece types (white and black pieces) plus an empty square. After applying a softmax over the class dimension, Fenify provides, for each square sq, a probability distribution: **Pr(piece｜frame t,sq)**

This representation preserves uncertainty at the square level and allows robust comparison against candidate board states, rather than relying on a single hard prediction.

PGN Parsing and FEN Normalization

From the PGN file, we extract the full sequence of board positions {P0,P1,…,PN}, where each position corresponds to a move (or ply) in the game. Each PGN position is converted into a canonical FEN representation and then expanded into a 64 square board labeling, where each square is assigned a single “true” class (one of the same 13 classes used by Fenify).

Importantly, non visual components of the FEN string (such as side to move, castling rights, en passant square, and move counters) are ignored. Only the piece placement field is used, ensuring compatibility with Fenify’s square wise predictions.

Distance Metric Between Frames and PGN Positions

To quantify how well a video frame **t** matches a candidate PGN position **Pi**, we define a cost function based on negative log likelihood, rather than string based distances such as Levenshtein distance. Specifically, the cost is computed as:

$$cost(t,\ i)\ =\ -\sum_{sq=0}^{63} logPr(piece\ at\ sq\ in\ P_i \mid frame\ t)$$

In other words, this metric sums the negative log probabilities that Fenify assigns to the correct PGN piece on each square. A lower cost indicates higher agreement between the visual evidence in the frame and the board configuration described by the PGN.

This formulation has several advantages:

- It uses the full probabilistic output of Fenify rather than a hard prediction.
- Errors on visually ambiguous squares contribute softly rather than catastrophically.
- The metric naturally aggregates evidence across all squares.

For example, if Fenify strongly predicts a white rook on a1 and a black king on e8, and the PGN position contains exactly those pieces, the corresponding squares will contribute very low cost terms. Conversely, a mismatch (e.g., predicting an empty square where the PGN has a queen) results in a high penalty.

Handling Camera Orientation via Symmetry Evaluation

Since chess videos may be recorded from different viewpoints, the mapping between image coordinates and board squares may be rotated or mirrored. To handle this, we evaluate all 8 dihedral symmetries of the board (four rotations and their mirrored versions). For each symmetry, we compute the cost as above and retain the minimum cost across symmetries. This ensures robustness to camera orientation without requiring explicit calibration.

Sequence Alignment via Dynamic Programming

We are given two ordered sequences:

- Video frames in temporal order: $F_0, F_1, \ldots, F_T$
- PGN positions in move order: $P_0, P_1, \ldots, P_N$

First, we construct a cost matrix where each entry **(t,i)** stores the cost of matching frame **Ft** to PGN position **Pi**. Empirically, low cost entries tend to form a diagonal structure, reflecting the fact that later frames correspond to later moves in the game.

Next, we compute the optimal alignment path using dynamic programming under a Next, we compute the optimal alignment path using dynamic programming under a monotonicity constraint:

- Each frame must be assigned exactly one PGN position.
- The PGN index may stay the same (multiple frames per position) or advance by one or more steps.
- Backward moves in the PGN index are forbidden.

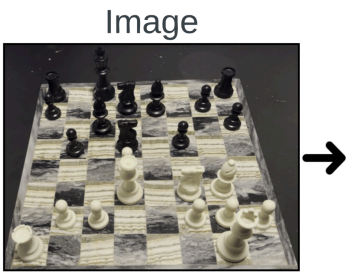
This constraint reflects the physical reality of a chess game: moves occur sequentially, and the board state never reverts to an earlier configuration.

Visually, the optimal path progresses only downward (next frame) or rightward (next PGN position) through the cost matrix. The resulting alignment function **a(t)** assigns each frame **t** to a PGN index i.

Final Label Assignment

Finally, each frame **t** is labeled with the FEN representation of the aligned PGN position **Pa(t)**. This produces a temporally dense, automatically labeled dataset in which every frame is associated with a high confidence board state derived from the original PGN.

This approach enables scalable dataset construction without manual annotation, while preserving precise alignment between video frames and symbolic game representations.



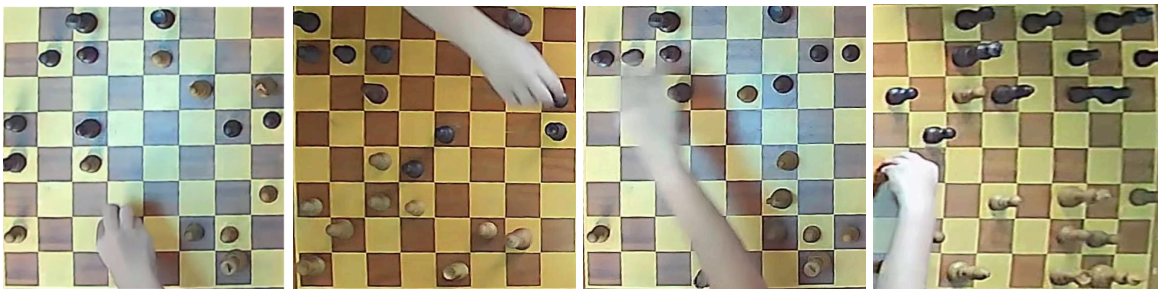
Metric	Value
Loss	3.88
Binary Accuracy	0.999
Colors Accuracy	0.974
Color-blind Accuracy	0.974
Full Accuracy	**0.950**

Fenify Accuracy

4.5.2 Dataset Filtering using MediaPipe Hand Detection

Real chessboard images and video frames often contain hands or partial occlusions caused by players interacting with the board. Such occlusions introduce visual noise and inconsistencies that are not present in the synthetic data and can negatively affect the stability and convergence of adversarial training. To reduce this source of domain mismatch, an automated filtering step was applied to remove samples containing visible hands.

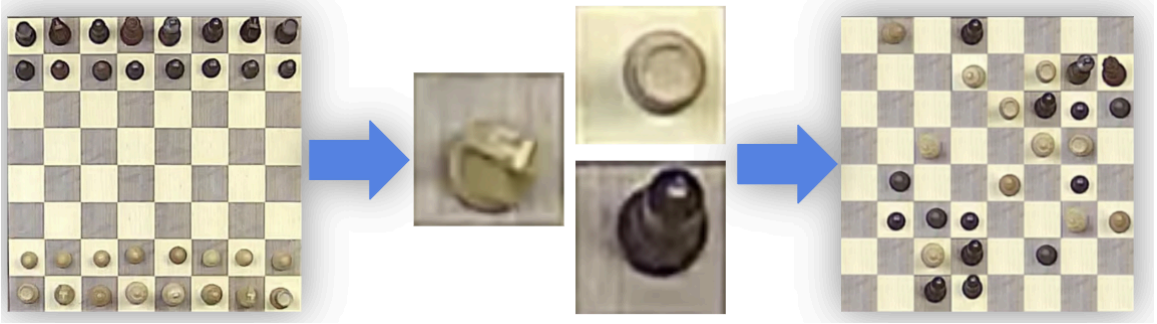
Hand detection was performed using a pre trained hand landmark detection model from Google MediaPipe framework. Each image was analyzed independently, and frames in which hands were detected were excluded from the dataset. This filtering step results in cleaner training pairs and improves the visual consistency between the synthetic and real domains.



4.5.3 Real Data Expansion via Board Level Recomposition

To further increase the diversity and size of the real image dataset, an additional data expansion strategy based on board level recomposition was employed. A small set of high quality overhead images of real chessboards was used as a source of square level visual components. Each board image was partitioned into individual square tiles, capturing both empty squares and squares containing pieces under real lighting and texture conditions.

New composite board images were then generated by rearranging these tiles to form novel board configurations. This process preserves realistic local appearance, including piece texture, shadows, and board material, while substantially increasing the number of distinct real images. The recomposed boards do not necessarily correspond to legal chess positions; however, they provide visually plausible samples that enrich the real domain and help the model generalize to a wider range of appearances.

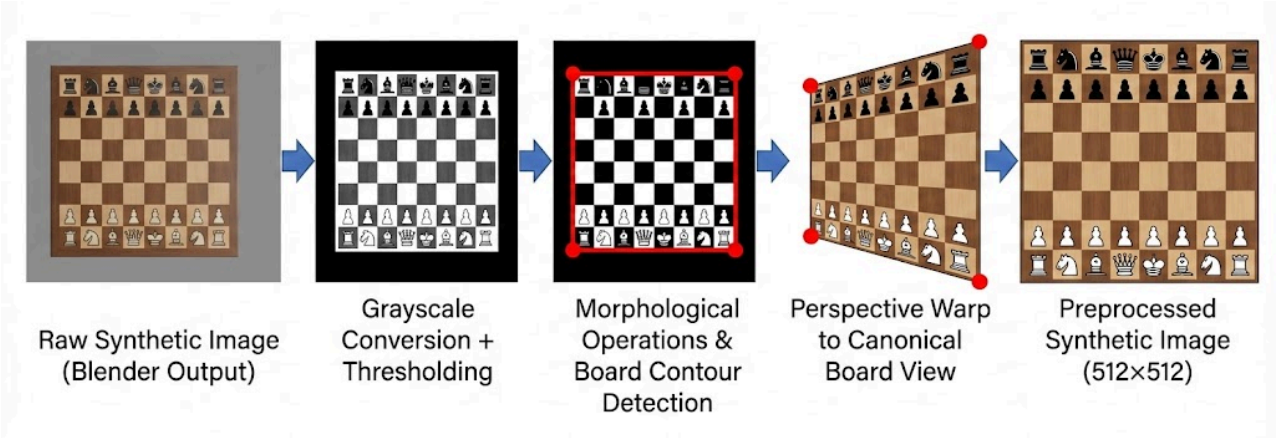


4.5.4 Synthetic and Real Image Preprocessing

Both synthetic and real images undergo preprocessing to ensure a consistent input format and to reduce domain discrepancies between the two sources. Synthetic

chessboard images generated using Blender often include large, uniform background regions surrounding the board. To address this, each synthetic image is processed using a board detection pipeline that identifies the chessboard boundaries via thresholding, contour detection, and geometric analysis. The detected board is then cropped, perspective corrected, and resized to a fixed resolution of 512×512 pixels.

Real images are processed using a simpler pipeline, as the chessboard is typically already well framed. These images are resized directly to the same target resolution without geometric warping. In both cases, images are converted to RGB format and normalized before being passed to the network, ensuring a unified representation for training.



4.5.5 Postprocessing and Visualization

After inference, the generator outputs are mapped from the network's output range back to standard RGB space to produce visually interpretable images. The resulting images are saved to disk for further analysis. For qualitative evaluation, side by side visualizations are generated, showing the synthetic input image alongside the corresponding translated output. These visual comparisons are used to assess visual realism, structural consistency of the chessboard, and the effectiveness of the synthetic to real translation.

5. Experiments

5.1 Dataset description and splits

The training dataset consists of paired synthetic real chessboard images organized by board position, represented using FEN notation. For each FEN entry, a single synthetic image generated from a graphics pipeline is paired with one or more corresponding real images captured from real world chess data. During training, one real image is randomly sampled per FEN instance at each iteration, introducing additional variability while preserving semantic alignment between the synthetic input and the real target.
The model is trained using the full available dataset. In addition to the originally labeled real image dataset, the dataset was expanded using two complementary sources: (1) additional labeled data derived from PGN files via automatic FEN extraction and alignment, contributing 284 samples. (2) board level recomposition of real images, which generated additional training examples. These expansions increase dataset diversity without introducing manual labeling.

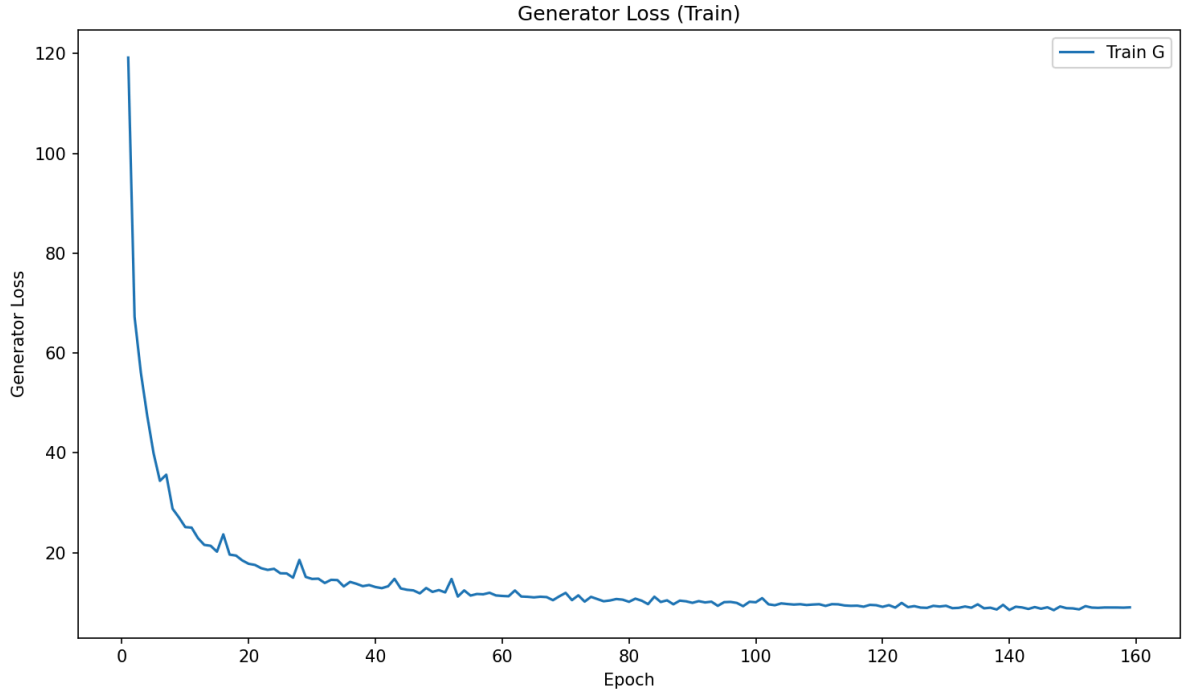
No explicit validation split is used; instead, the model is trained on the complete dataset. Evaluation is performed qualitatively using a fixed, held out synthetic test set. This test set is not used during training and is processed periodically to monitor visual translation quality and training progression.

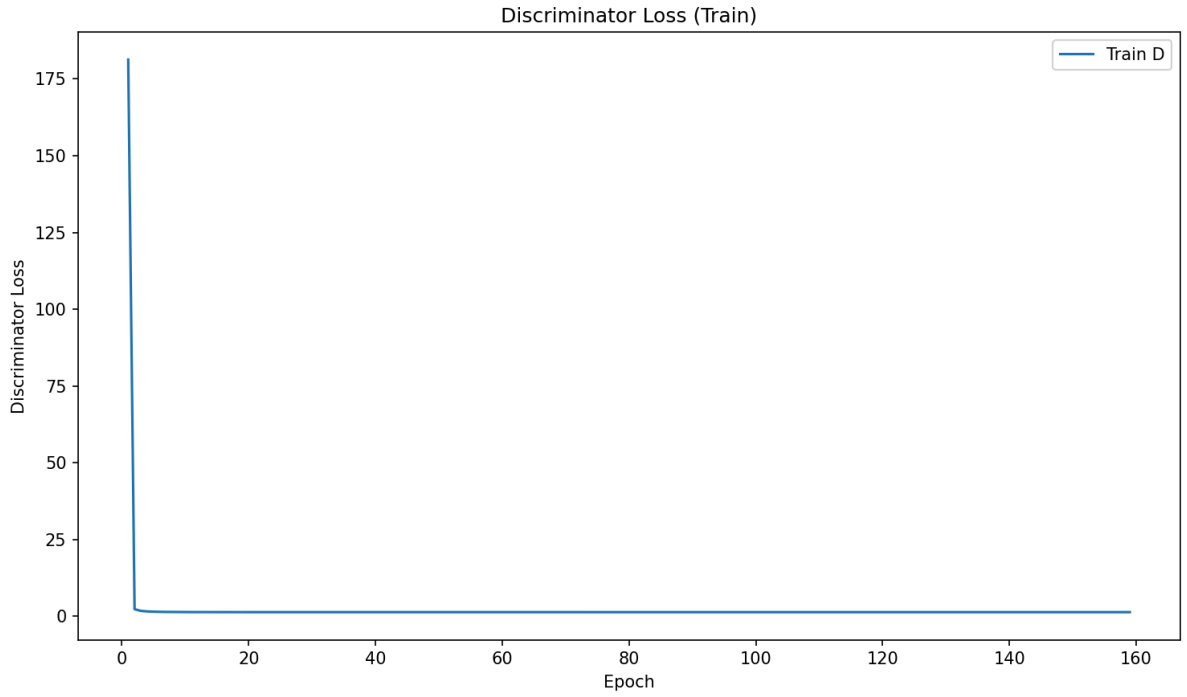
5.2 Evaluation Metrics

Model evaluation is performed using internal training signals and qualitative visual inspection rather than external image quality metrics. During training, the generator and discriminator losses are monitored across epochs to assess convergence behavior and overall training dynamics. The generator loss is composed of an adversarial component, an L1 reconstruction loss, and a perceptual loss based on deep feature differences, while the discriminator loss reflects its ability to distinguish real from generated images. These loss values are logged throughout training and used to track optimization progress.

In addition to loss monitoring, qualitative evaluation is conducted by visually inspecting generated images produced on a fixed, held out synthetic test set. Side by side comparisons between the synthetic inputs and their translated outputs are used

to assess visual realism, structural consistency of the chessboard, and preservation of semantic layout.





5.3 Baselines and Comparisons

No external baseline models are used for direct quantitative comparison in this work. Instead, evaluation is conducted through internal comparisons across different training configurations and across training time. In particular, the model's behavior is compared at different stages of training by analyzing generator outputs saved at regular intervals. This allows observation of the progression from early, low quality translations to more realistic and visually coherent outputs in later epochs.

Additional comparisons are implicitly performed by varying key training configurations, such as the choice of adversarial loss function and the strength of the discriminator. These comparisons provide insight into how architectural and optimization choices influence training stability and output quality. The results of these comparisons are reflected in both the observed loss dynamics and the qualitative visual outputs, which are analyzed in subsequent sections.

5.4 Quantitative Results

This work focuses primarily on visual realism and perceptual quality in a synthetic to real image translation setting, where no exact ground truth real image exists for each synthetic input. As a result, standard quantitative image similarity metrics are not directly applicable or meaningful for evaluation. Consequently, no numerical performance metrics are reported for output image quality.

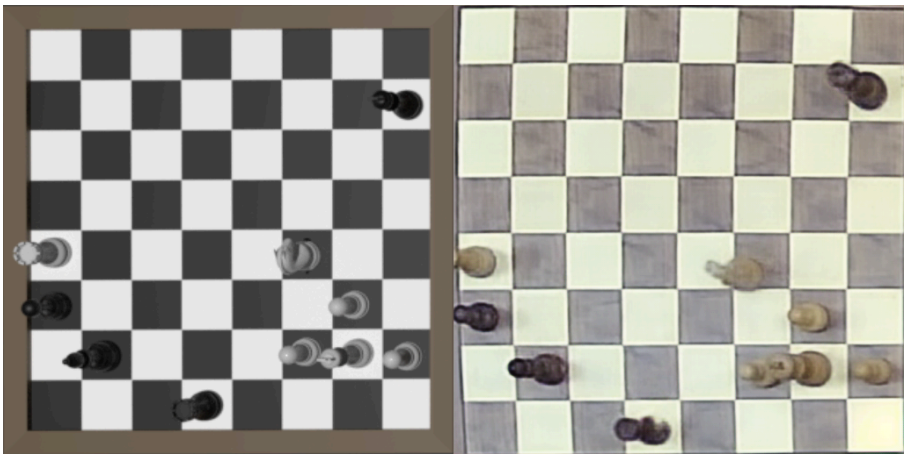
Quantitative analysis is limited to monitoring training dynamics through the generator and discriminator loss curves presented in Section 5.2. These loss values are used solely as indicators of optimization behavior and training stability, rather than as measures of final output quality. The evaluation of the proposed method therefore relies mainly on qualitative visual results, which are presented in the following section.

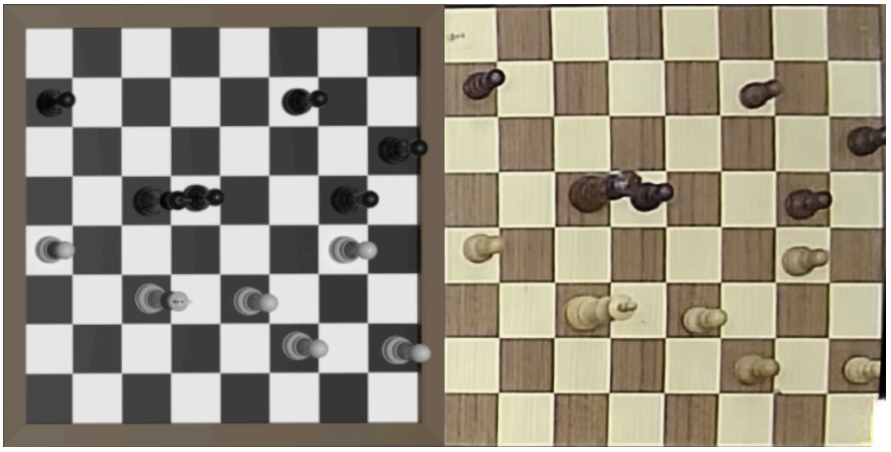
5.5 Qualitative Results (Visualizations)

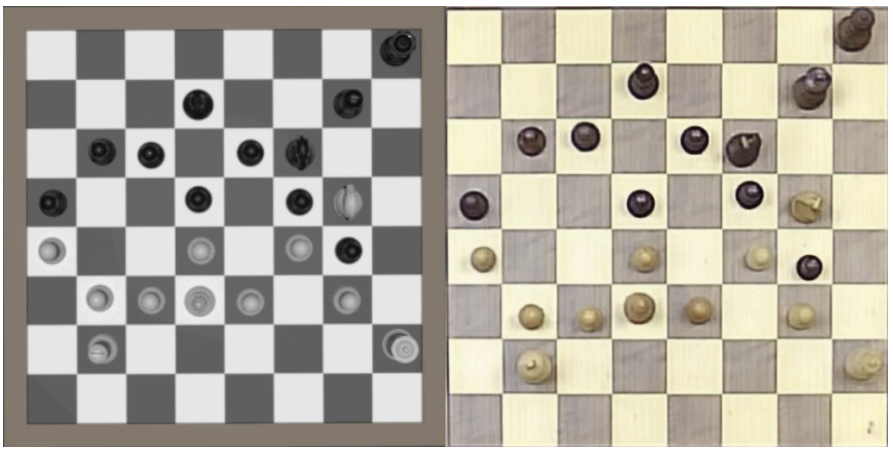
5.5.1 Synthetic Input to Generated Output

The figures present side by side comparisons between synthetic chessboard inputs and the corresponding images generated by the proposed model. Each example illustrates the model's ability to translate synthetic renderings into visually realistic chessboard images while preserving the underlying board structure and piece layout. The generated outputs exhibit improved texture quality, more natural lighting, and reduced synthetic artifacts compared to the input images.

Across the presented examples, the spatial arrangement of squares and the placement of chess pieces remain consistent with the synthetic inputs, indicating that the translation process maintains semantic correctness. At the same time, the model successfully introduces visual characteristics typical of real world chessboard images, demonstrating effective synthetic to real domain adaptation.



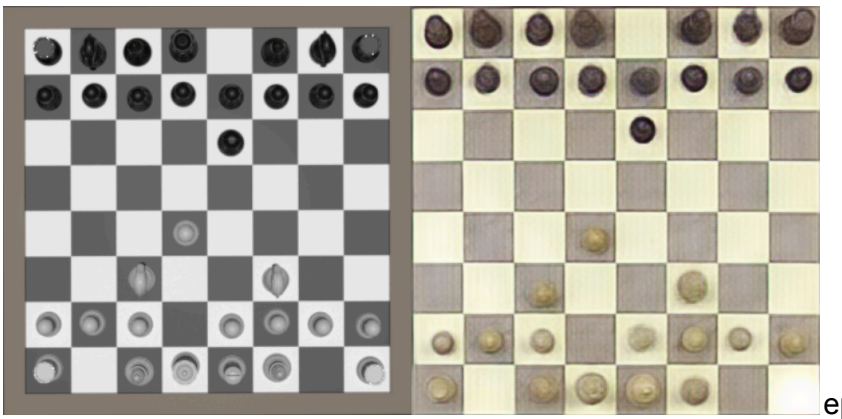


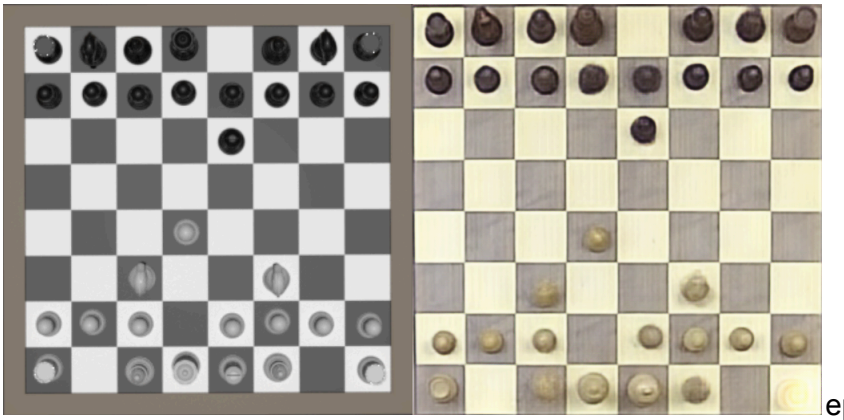


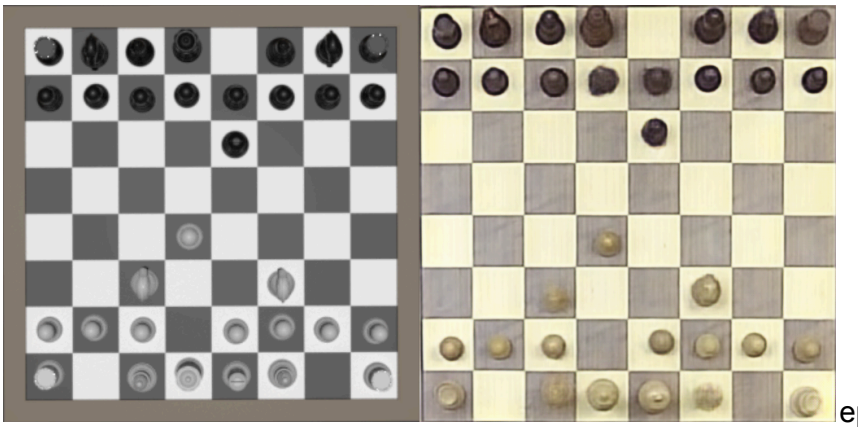
5.5.2 Progression Across Training Epochs

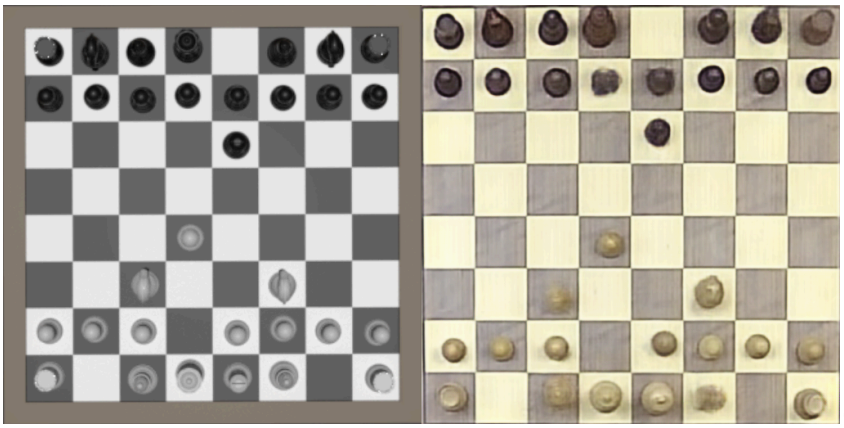
Figure X illustrates the progression of the generated outputs across different stages of training for a fixed synthetic input. The figure presents side by side results produced by the generator at epochs 10, 50, 100, and 150, allowing direct visual comparison of the model's evolution over time.

At early training stages (epoch 10), the generated images exhibit limited realism, with blurred textures and incomplete visual details. As training progresses to epochs 50 and 100, the outputs become increasingly coherent, showing improved texture smoothness, clearer square boundaries, and more realistic appearance of board materials. By epoch 150, the generated images display stable visual structure and enhanced realism, with fewer artifacts and more consistent lighting and texture patterns. These results visually demonstrate the gradual refinement achieved through adversarial training and highlight the model's ability to progressively improve output quality as training advances.


epoch 10


epoch 50


epoch 100

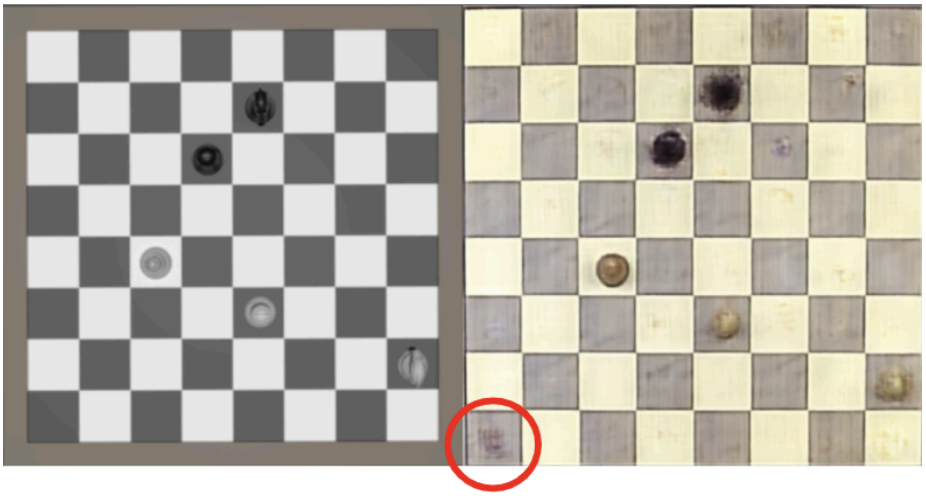

epoch 150

6. Ablation Study

To assess the contribution of each major component in the proposed framework, we conduct a series of ablation experiments in which individual loss terms or model components are removed while keeping all other settings unchanged. The effects of these modifications are evaluated qualitatively through visual inspection of the generated outputs.

6.1. Removing the Perceptual (VGG) Loss

When the perceptual loss is removed from the training objective, the generator continues to produce structurally plausible chessboards; however, the visual quality of the outputs degrades noticeably. In particular, the model begins to introduce blurry, cloud like artifacts in regions of the board where no pieces are present. These hallucinated textures reduce the visual coherence of empty squares and negatively affect the realism of the generated images. This behavior indicates that the perceptual loss plays a critical role in preserving high level visual consistency and suppressing unnatural artifacts in textureless regions.

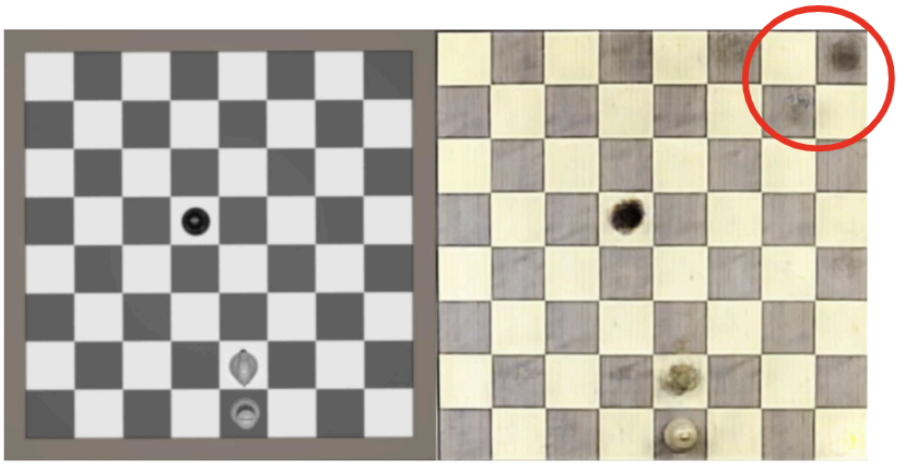


6.2 Removing the L1 Reconstruction Loss

Eliminating the L1 reconstruction loss leads to a significant degradation in local texture quality. While the global structure of the board remains largely intact, close inspection of individual squares reveals highly pixelated regions, especially when zooming into areas such as wooden board textures. The generated images lack the smooth, continuous appearance observed in real chessboard surfaces. This demonstrates that the L1 loss is essential for enforcing fine grained, low frequency accuracy and maintaining realistic local textures.

6.3 Removing the Discriminator

Removing the discriminator eliminates the adversarial training signal and reduces the model to a purely reconstruction based objective. In this setting, the generator exhibits a strong tendency to hallucinate chess pieces in empty squares, producing visually implausible board states. Similar to the case without perceptual loss, these hallucinations introduce artifacts that violate the semantic structure of the board. This result highlights the importance of the discriminator in enforcing realism and preventing the generation of implausible visual content.



7. Discussion and Limitations

While the proposed synthetic to real translation framework produces visually convincing results, several limitations remain. One notable failure mode occurs in challenging board configurations or regions with subtle textures, where the generator may introduce minor artifacts or inconsistencies. These issues are more pronounced in cases involving complex lighting patterns or uncommon visual layouts that are underrepresented in the training data.

A key limitation of the current approach is the imbalance between the generator and the discriminator during training. In our experiments, the discriminator often

dominates the adversarial game, effectively overpowering the generator and limiting its ability to further improve visual realism. This imbalance can hinder convergence and restrict the generator’s capacity to refine fine grained details, particularly in later training stages.

Future work should focus on improving the balance between the generator and discriminator. Potential directions include adaptive adjustment of discriminator strength, alternative adversarial objectives, dynamic loss weighting, or training schedules that regulate discriminator updates. Additional improvements could be achieved by incorporating larger and more diverse real world datasets, as well as more advanced augmentation strategies. Addressing these limitations would likely lead to more stable training dynamics and higher quality synthetic to real translations.

References

[1] Isola et al., "Image-to-Image Translation with Conditional Adversarial Networks", CVPR 2017.

[2] Fenify repository / tool https://github.com/notnil/fenify-3D.

[3] python-chess: a chess library for Python
https://python-chess.readthedocs.io/en/latest/

[4] Google MediaPipe Hands repository
https://github.com/google-ai-edge/mediapipe/blob/master/docs/solutions/hands.md