

Towards improving handwritten chess-notation classification using machine learning models

Fiszbin, Vincent V. L., Bradicic, Manuel, Lanfranconi, Michele
Project 2, CS-433, EPFL, Switzerland

Abstract—Accurate character recognition is crucial for applications such as handwriting classification and chess notation interpretation. This study presents a comparative analysis of Bidirectional Long Short-Term memory (BiLSTM) and Vision Transformer (ViT) models trained on synthetic data. Utilising statistical metrics, the BiLSTM model demonstrated superior performance, particularly in distinguishing visually similar characters. These findings highlight BiLSTM’s effectiveness in capturing sequential dependencies and contextual information, showcasing its architecture’s suitability to this challenge. Conversely, while ViT exhibits robust feature extraction capabilities in general, it requires further refinement and enhancement to perform better on this problem.

I. INTRODUCTION

When chess players compete at tournaments, they write down their moves on a chess scoresheet by hand. After the game, these hand-written documents have to be digitized for further processing and analysis. This is usually done by hand, by typing all chess moves into the computer. One way to simplify this task is the use of Optical Character Recognition (OCR) engines to extract the handwritten chess moves from a picture of the score-sheet. The aim of the project was to improve the current system (available for testing at <https://chessreader.org>).

Motivated by the recent success of Vision Transformer (ViT) [1] in computer vision tasks has motivated us to explore its potential in chess notation recognition. However, it is known that ViT does not introduce strong inductive bias and requires vast quantities of annotated training data to perform well. Therefore, in addition to ViT, we explore the use of CNN-BiLSTM, which has proven effective in text recognition tasks due to its ability to process variable-length sequences and capture contextual dependencies. We combine synthetic data generation with advanced architectures to evaluate their performance in accurately recognising handwritten chess notations. We outline different strategies to address this problem, including the creation of synthetic training data and integration of domain-specific knowledge into the model architectures.

An introduction to chess notations is provided in **Section II**. **Section III** depicts the methodology underlying the synthetic data generation process, as well as the BiLSTM and ViT based approach, developed to address the problem. The experimental setup and results are presented in **Section IV**, and the discussion of obtained results is discussed

in **Section V**. Finally, the study is concluded in **Section VI**, following a discussion on ethical risks and concerns in **Section VII**.

II. CHESS NOTATION

Unlike classical OCR tasks, our character set is limited to standardized chess notation, including lowercase letters (a-h), numbers (1-8), piece abbreviations (R, N, B, Q, K), and symbols for castling (O, -), promotion (=), capture (x), check (+), and checkmate (#), totaling 27 characters. This restricted vocabulary reduces the scope of recognition and minimizes potential confusion, such as mistaking “S” for “5,” which are absent in our dataset.

III. METHODOLOGY

In this section, we first discuss the mechanism for generating synthetic data. Subsequently, we describe the implementation of two models: CNN-BiLSTM and ViT.

A. Synthetic Data Generation

The first part of our project was to prepare a set of data to train our model. Due to the small amount of labeled data that we had, we could not directly train a model, and instead we had to generate some synthetic data ourselves. To do so, we have reimplemented existing pipeline, to which we applied several improvements.

The main idea is to generate image containing texts and boxes which are similar to those in chess score-sheets, by sampling a random chess move and then using one of over 360 fonts which simulate hand-written text to draw such text on an image. Either full-lines or dashed-lines were implemented, with additional text pieces of text were written around the box to stimulate real data.

Given the nature of the chess game and with the setup of the pipeline, there exists more than 11 million different possible images ($16000+$ moves \times $360+$ fonts \times 2 linestyles for the boxes $\approx 11.5 \times 10^6$).

In addition, several post-processing steps were employed to increase robustness of the models; Geometric transformations (rotation, translation, shearing), perspective effect, zoom and brightness, noise and blue, character change¹.

¹Since not all ASCII characters are part of the class vocabulary, and in order to increase the robustness of models, during the image generation we substituted some characters with other similar ones (such as **1** with **l**, and **s** with **5**)

B. Convolutional Neural Network (CNN) and Bidirectional Long Short-Term Memory (BiLSTM) architecture

We adapted a model architecture showcased in the open-source MLTU library [2], which is designed for handwritten text recognition. The image input is sent through convolutional layers, composed of residual blocks (See Figure 4) inspired by ResNet [3], which mitigate the vanishing gradient problem and enable the training of deeper architectures.

The feature maps extracted by the convolutional layers are fed into a BiLSTM, regarded as a powerful tool for handwriting recognition [4]. LSTMs process variable-length sequences, which is required for our case, as chess notation moves vary in length. To make predictions, BiLSTM leverages contextual information from the entire input, using both preceding and following characters. To optimise the model, the Connectionist Temporal Classification (CTC) loss function [5] is employed. The BiLSTM output is passed to a fully connected layer to produce a probability matrix over the allowed characters in the vocabulary (27 characters plus 1 blank token). CTC loss is then applied to align the predicted sequence with the true label. Finally, CTC decoder was used to generate the most likely character sequence. CTC is well-suited for sequences with variable lengths as it provides a structured way to learn the mapping between true and predicted notation.

Character Error Rate (CER) is employed as the primary evaluation metric to assess the performance of our model on test data. CER calculates the percentage of character-level errors, comprising insertions, deletions, and substitutions, between the predicted and ground truth sequence. Unlike the F1-score, which balances precision and recall, CER Offers a more detailed analysis of the model's accuracy on sequential data. Therefore, CER was utilised over F1-score for evaluating a performance of a model overall. Cross-validation was not conducted in this study because the synthetic data generation process ensures a sufficiently large and diverse dataset, rendering CV both redundant and computationally inefficient. Instead, the test dataset consisted of real-world chess notation images provided by the hosting laboratory.

C. Visual Transformer (ViT)

Conventional Transformer receives as an input a 1D sequence of token embeddings. To manage 2D images, we reshape an image of chess notations $x \in \mathbb{R}^{H \times W \times C}$ into a sequence of flattened 2D patches [1]. Essentially, image patches are treated the same way as token words in an NLP application, similar to BERT [6]. These patches are then passed into a stack of 12 transformer encoder layers, where each layer consists of a multi-head self-attention mechanism (MSA), a feed-forward network (FNN) that consists of an intermediate linear layer and GELU activation. Layer normalisation (LN) is placed both before and after attention

and FNN (Eq. 1, 2). Overview of the architecture is depicted in Appendix B.

$$y^n = x^{n-1} + MSA(LN(x^{n-1})) \quad (1)$$

$$x^n = y^n + FFN(LN(y^n)) \quad (2)$$

where x^{n-1} in the input of the n-th block.

Since ViT alone is not suited for handwritten text recognition [7] and considering the variable length of hand written notation, we provide an adjustment to the ViT, introducing a novel approach to address this challenge. The output of the encoder is fed into a series of seven parallel linear classifier heads (Eq. 3). Each head independently maps the final pooled embedding to one of the 28 classes by applying softmax and choosing an index z_i with the highest probability (Eq. 4). Finally, we use index z_i to retrieve the corresponding character c_i from the vocabulary (Eq. 5), allowing the model to produce a sequence of seven character predictions from a single chess-notation image.

$$\mathcal{V} = \{c_1, c_2, \dots, c_{28}\}^\dagger$$

$$Y = Wx + b, \quad Y \in \mathbb{R}^7 \quad (3)$$

$$z_i = \max(\text{softmax}(\vec{y}_i)), \quad \text{for } Y = (\vec{y}_1, \dots, \vec{y}_7) \quad (4)$$

$$c_i = \mathcal{V}[z_i] \quad (5)$$

In addition to the architectural adjustments, AdamW optimizer and CTC loss are used (see III-B for more details). AdamW [8] introduces a decoupled weight decay, which helps maintain generalisation capabilities while preventing overfitting. Considering our large pool of training data and diverse image-based tokens it fits our needs.

IV. EXPERIMENTS

A. Experimental Setup

The Adam optimizer was used to train the CNN-BiLSTM network, with a learning rate starting at 5×10^{-3} and decreasing by 10% whenever the validation CER plateaued for 10 consecutive epochs, with a minimum bound of 1e-6. The images were resized to $3 \times 32 \times 128$. The batch size was set to 64 and image.

We used Google's pretrained ViT weights as the backbone of our model due to limited computational resources. We then fine-tuned all ViT layers on our generated dataset with a varying learning rate; 1×10^{-4} proved working the best (see Appendix H). In addition dataloader processed batches of 32 and 64 images, each resized to $3 \times 224 \times 224$ by using padding (See appendix x sample). Additionally, images were normalised using $mean = [0.485, 0.456, 0.406]$ and $std = [0.229, 0.224, 0.225]$, consistent with ImageNet

[†]The chess notation vocabulary consists of 27 characters. However, the vocabulary dictionary includes an empty space, bringing the total to 28 classes

preprocessing. Normalising images proved to be a crucial step in obtaining higher accuracies.

Different dataset sizes were tested when training the CNN-BiLSTM network, including: 10^4 , 10^5 , 10^6 synthetic images. The 100,000 size yielded the best accuracy, and increasing dataset size did not improve performance. Due to computational limitations, we did not conduct tests on larger datasets with the ViT model. All subsequent experiments were performed on 100,000 synthetic images. Lastly, all the transformations applied to these images were added stochastically with 20 % probability.

The models were trained on an NVIDIA Tesla V100-SXM2-32GB GPU provided by the Research Computing Platform at EPFL. On average, training a ViT take 16 min per epoch, and was finetuned on 10 epochs. Similarly CNN-BiLSTM network takes slightly over 1 min per epoch and was trained on 100 epochs.

B. Quantitative Analysis

To assess the performance of the BiLSTM and ViT models, we analyse CER as the primary evaluation metric on notation level. To evaluate performance across classes (i.e. characters) we focus on precision recall and f1-score.

Overall, the CNN-BiLSTM demonstrates superior performance compared to the ViT model on the given problem, which is depicted in Table I. The same is evident in the Figures 1 and 2.

The CNN-BiLSTM model consistently outperformed the ViT model across most character classes (See Appendix D, Table on CNN-BiLSTM Metrics). This is evident in higher precision, recall and f1-score across almost all characters, indicating that CNN-BiLSTM not only identifies true positives, but also effectively minimises false positive and false negatives. For example, characters **N**, **Q**, **R**, and **x** achieved nearly perfect F1-scores of 0.97, 0.94, 0.90 and 0.95 respectively. On the other side, ViT model demonstrated much lower performance across majority of the classes, it generally lagged behind CNN-BiLSTM. Notably, the ViT model exhibited lower precision scores for classes like **+**, **-** and **=** (See Appendix D, Table on ViT Metrics). This suggests the flaws of the ViT attention mechanism compared to CNN-BiLSTM’s convolutional layers which were able to learn this geometrical representations of different lines.

In addition, Table II highlights the most frequent misclassifications made by the CNN-BiLSTM model. The model displayed a tendency to confuse characters with similar visual features.

C. Qualitative Analysis

To obtain a more clear insights into the performance of the two models, a qualitative analysis was conducted using confusion matrices for each class. The results reveal that the CNN-BiLSTM model outperforms the ViT model in terms of overall accuracy for the given task - as discussed above.

Model Name	CER	Test set accuracy (%)
CNN-BiLSTM	0.113	74.96
ViT	0.604	62.38
Google	-	67.47
Azure	-	52.10
Amazon	-	47.38
Abby	-	80.13

Table I: Performance of SOTA models on the test dataset

Label	Prediction	Count	Occurrence [%]
a3	g3	14	4.19
h6	b6	13	3.89
Na2	Na3	12	3.59
h5	b5	12	3.59
h4	b4	10	2.99
e4	c4	8	2.40
Rb8	Bb8	7	2.10
Ng4	Bg4	7	2.10
Kb1	Kb4	7	2.10
c5	f5	5	1.50

Table II: Most common missclassifications of the CNN-BiLSTM model; The CNN-BiLSTM model frequently confuses characters like **a3** with **g3**, **h6** with **b6**, and **e3** with **c4**. These misclassifications are caused most likely because of the visual and structural similarities between these notations.

The superior performance of CNN-BiLSTM showcases its enhanced capability in capturing sequential dependencies and contextual information, which are crucial elements for classifying characters such as handwriting recognition.

Examining the confusion matrix for the ViT model and CNN-BiLSTM (See Appendix C), it becomes clear that the ViT model produces higher miss-classification of characters (expressed with stronger heat off the diagonal), compared to CNN-BiLSTM. In addition, it becomes clear that the models encounter difficulties in correctly identifying certain characters. For instance, they struggle to differentiate between the characters **e** and **c**, as well as **b** and **d**. These misclassifications most likely stem from the visual similarities between these characters, posing a challenge for model’s feature extraction mechanism. See Appendix F for examples of misclassified, visually ambiguous images.

D. Ablation Study

An ablation study was conducted (see Figure III) with the CNN-BiLSTM network to evaluate the impact of different data pipeline components on the final accuracy achieved on the test set.

V. DISCUSSION

The CNN-BiLSTM outperforms the ViT on real-world example images in the test set. This better performance can be attributed to the convolutional layers of the CNN-BiLSTM, which are better suited to extracting the handwritten patterns present in chess notation images. In contrast, the attention mechanism of the ViT (see Figure 9) may struggle

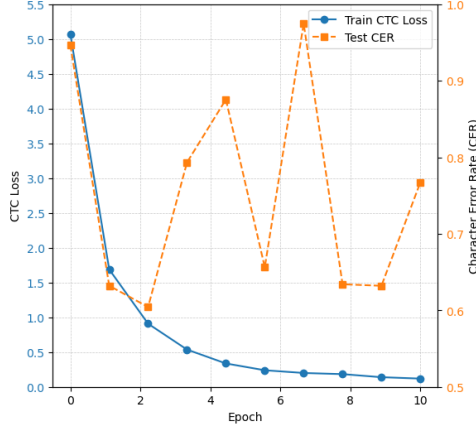


Figure 1: **ViT**; Training CTC Loss and Test CER curves for the ViT model over 10 epochs. Leveraging pretrained weights from the ImageNet dataset, the model achieves its lowest CER score of 0.604 by the second epoch. However, subsequent reductions in CTC loss are accompanied by overfitting, as evidenced by fluctuating CER values.

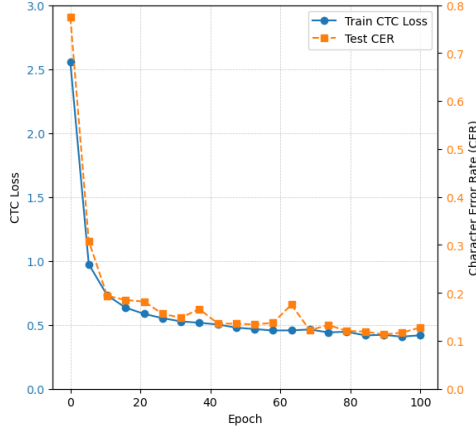


Figure 2: **CNN-BiLSTM**; Training CTC Loss and Test CER curves for the CNN-BiLSTM model across 100 epochs. The plot illustrates the convergence of the training process, showing a consistent decrease in loss and error rate, with CER stabilizing as the model learns to align predictions with the ground truth effectively.

to capture these patterns as effectively. Analysis of the confusion matrices and the most common mistakes confirms CNN-BiLSTM’s superiority, particularly in distinguishing visually similar characters.

CNN-BiLSTM outperforms the benchmark models: Google, Azure, and Amazon (see Table I), demonstrating the superiority of a model specifically trained on handwritten chess notation images compared to more general-purpose handwriting or text recognition engines. However, the Abbyy model still achieves better performance than the

Table III: Ablation Study Results; **Box**: Gridlines drawn on the generated chess move images to replicate the boxes found on chess scoresheets. **Processing**: The data processing pipeline described in III-A. **Brightness, Dilate, Sharpen, Rotate (BDSR)**: Additional data augmentations from the MLTU library applied to the training data. **Next Best Prediction**: This mechanism checks if the model’s prediction is a syntactically valid chess move. If not, the next most confident prediction is selected. This process repeats until a valid move is found or the prediction with the highest confidence is chosen by default. The multiple predictions and their confidence scores are obtained using beam search during the CTC decoding step.

Box	Processing	BDSR	Next Best Prediction	Accuracy (%)
✓	✓	✓	✓	74.96
✓	✓	✓		74.66
✓	✓			72.63
✓		✓		71.06
✓				64.24
		✓		18.37
				5.77

CNN-BiLSTM. A more refined data generation pipeline could potentially enable the custom CNN-BiLSTM model to surpass it.

The ablation study highlights that the Box component provides the most significant accuracy gain, followed by data processing and MLTU augmentations, indicating that additional data augmentation improves generalization on the test set. The Next Best Prediction component slightly increased accuracy by effectively handling invalid move predictions and selecting a better alternative from the models’s output.

Finally, increasing the synthetic dataset size beyond 100,000 images does not improve accuracy, likely because synthetic data is not the best representation of real data. For the same reason, increase in train/val loss doesn’t reflect performance on test. This plateau suggests limitations in the synthetic data’s diversity, underscoring the need for more realistic data generation to enhance generalization to real-world examples.

VI. CONCLUSION

This study highlights the superior performance of the CNN-BiLSTM model compared to ViT and general-purpose engines for handwritten chess notation recognition. By leveraging convolutional layers and sequential learning, the CNN-BiLSTM proves effective at recognizing handwritten chess moves. The findings also emphasize the critical role of data augmentation in achieving robust generalization to real-world data and suggest that refining the data generation process presents a clear path for further performance improvements. Lastly, they demonstrate the flaws of attention mechanism compared to CNN in hand written notation.

VII. ETHICAL RISKS AND CONCERNS

In analyzing the task we have been working with and developing machine learning architectures for recognizing handwritten chess scoresheets, several ethical concerns were identified and addressed. The analysis was mainly done through a literature review and theoretical analysis, since the test data we were given didn't allow for evaluation through fairness metrics such as Independence, Separation and Sufficiency.

First of all, the field that this project aims to **empower** is very narrow due to the limited application of our model, that is the field of chess tournaments/chess analysts, for whom we tried making it easier to digitize scoresheets accurately, streamlining workflows, and reducing the need for manual transcription.

A first point to address is that of **privacy**: while recorded moves of a chess match are not sensible information, the score sheets themselves include some information regarding the players such as their names, dates... In the current system, the approach to avoid the transfer of sensitive information is associated with the fact that only the boxes with the moves are passed to ML models, but not relying on cloud services adds a further layer of security.

To maximize the **fairness** of our system we entirely used synthetic data for our training, using several handwriting fonts in order to exclude any possible bias in the training of our model associated with data coming from only a subset of the future applicable customers.

Finally, we want to point out that the complete system involves a post-ML human check of the results predicted by the model. This ensures that the model will not lead to any harmful errors, hence upholding the principle of **non-maleficence**.

REFERENCES

- [1] A. Dosovitskiy, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.
- [2] R. Liuberskis, "Machine learning training utilities (for tensorflow and pytorch)." [Online]. Available: <https://github.com/pythonlessons/mltu/>
- [3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.
- [4] N. Majid and O. Eicher, "Digitization of handwritten chess scoresheets with a bilstm network," *J. Imaging*, vol. 8, no. 2, p. 31, January 2022.
- [5] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 369–376.
- [6] J. Devlin, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.
- [7] V. Agrawal, J. Jagtap, and M. P. Kantipudi, "Decoded-vit: A vision transformer framework for handwritten digit string recognition," *Revue d'Intelligence Artificielle*, vol. 38, no. 2, p. 523, 2024.
- [8] I. Loshchilov, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.
- [9] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.
- [10] O. F. Ayilara, L. Zhang, T. T. Sajobi, R. Sawatzky, E. Bohm, and L. M. Lix, "Impact of missing data on bias and precision when estimating change in patient-reported outcomes from a clinical registry," *Health and quality of life outcomes*, vol. 17, pp. 1–9, 2019.
- [11] Y. Zhang, E. Vittinghoff, M. J. Pletcher, N. B. Allen, A. Zeki Al Hazzouri, K. Yaffe, P. P. Balte, A. Alonso, A. B. Newman, D. G. Ives *et al.*, "Associations of blood pressure and cholesterol levels during young adulthood with later cardiovascular events," *Journal of the American college of cardiology*, vol. 74, no. 3, pp. 330–341, 2019.
- [12] J. R. Van Ginkel, M. Linting, R. C. Rippe, and A. Van Der Voort, "Rebutting existing misconceptions about multiple imputation as a method for handling missing data," *Journal of personality assessment*, vol. 102, no. 3, pp. 297–308, 2020.
- [13] R. H. Kallet, "How to write the methods section of a research paper," *Respiratory Care*, vol. 49, no. 10, pp. 1229–1232, 2004.
- [14] G. Anderson, "How to write a paper in scientific journal style and format," 2004, <http://abacus.bates.edu/ganderso/biology/resources/writing/HTWtoc.html>.
- [15] S. P. Jones, "How to write a great research paper," 2008, microsoft Research Cambridge.
- [16] Editorial, "Scientific writing 101," *Nature Structural & Molecular Biology*, vol. 17, p. 139, 2010.
- [17] J. B. Buckheit and D. L. Donoho, "Wavelab and reproducible research," Stanford University, Tech. Rep., 2009.
- [18] R. Gentleman, "Reproducible research: A bioinformatics case study," *Statistical Applications in Genetics and Molecular Biology*, vol. 4, no. 1, 2005. [Online]. Available: <http://www.bepress.com/sagmb/vol4/iss1/art2>
- [19] M. Schwab, M. Karrenbach, and J. Claerbout, "Making scientific computations reproducible," *Computing in Science and Engg.*, vol. 2, no. 6, pp. 61–67, 2000.
- [20] J. Spolsky, *Joel on Software: And on Diverse & Occasionally Related Matters That Will Prove of Interest etc.: And on Diverse and Occasionally Related Matters ... or Ill-Luck, Work with Them in Some Capacity*. APRESS, 2004.
- [21] A. Hunt and D. Thomas, *The Pragmatic Programmer*. Addison Wesley, 1999.

APPENDIX

A. BiLSTM Architecture

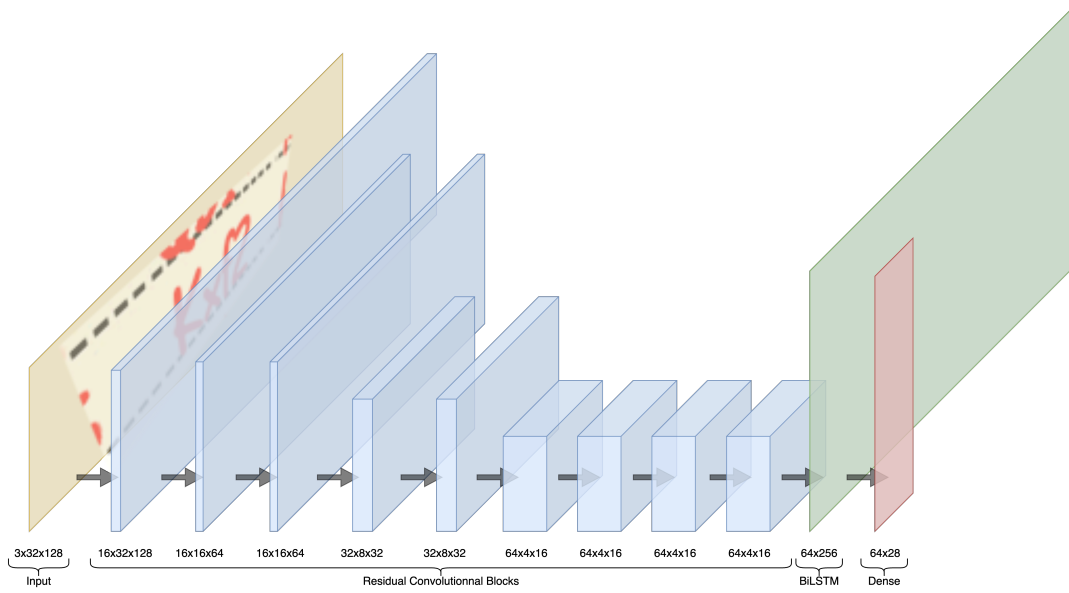


Figure 3: Model architecture of the convolutional BiLSTM network. Batch normalization, LeakyReLU, and Dropout are not shown.

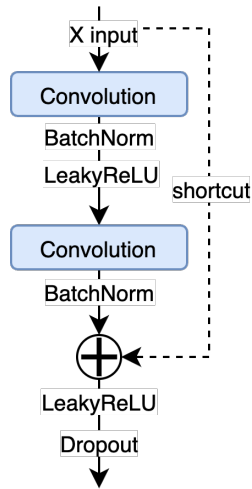


Figure 4: Residual block of CNN-BiLSTM

B. ViT Architecture

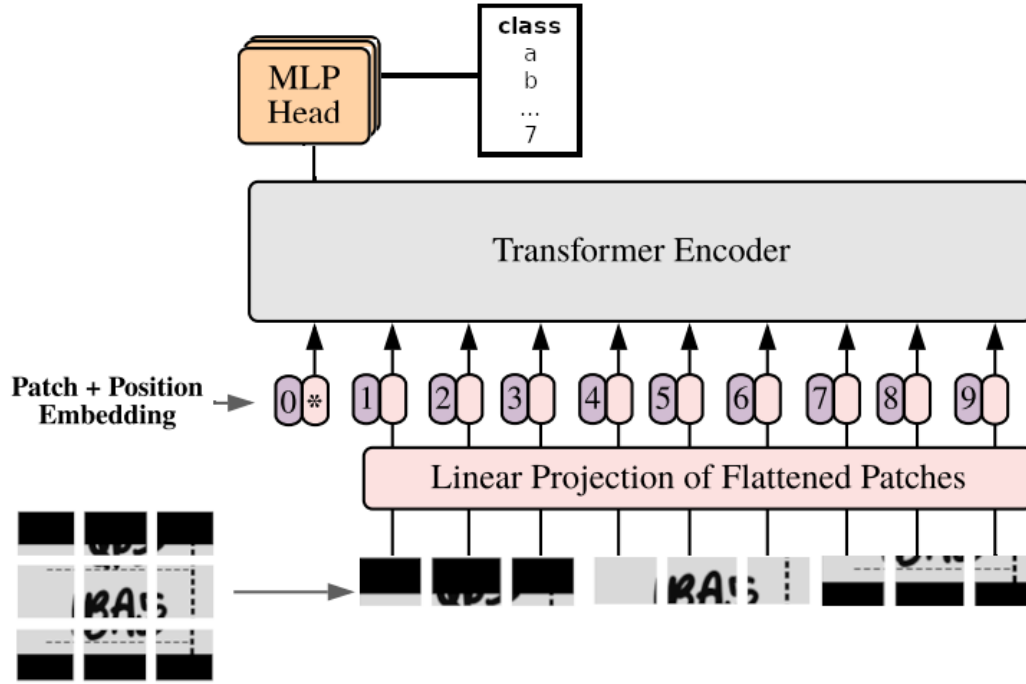


Figure 5: Architecture overview; A standard ViT-based framework augmented with an MLP head that produces seven character predictions from a 28-class vocabulary, each determined by a dedicated softmax output. Figure inspired by [1].

C. Confusion matrix of character predictions

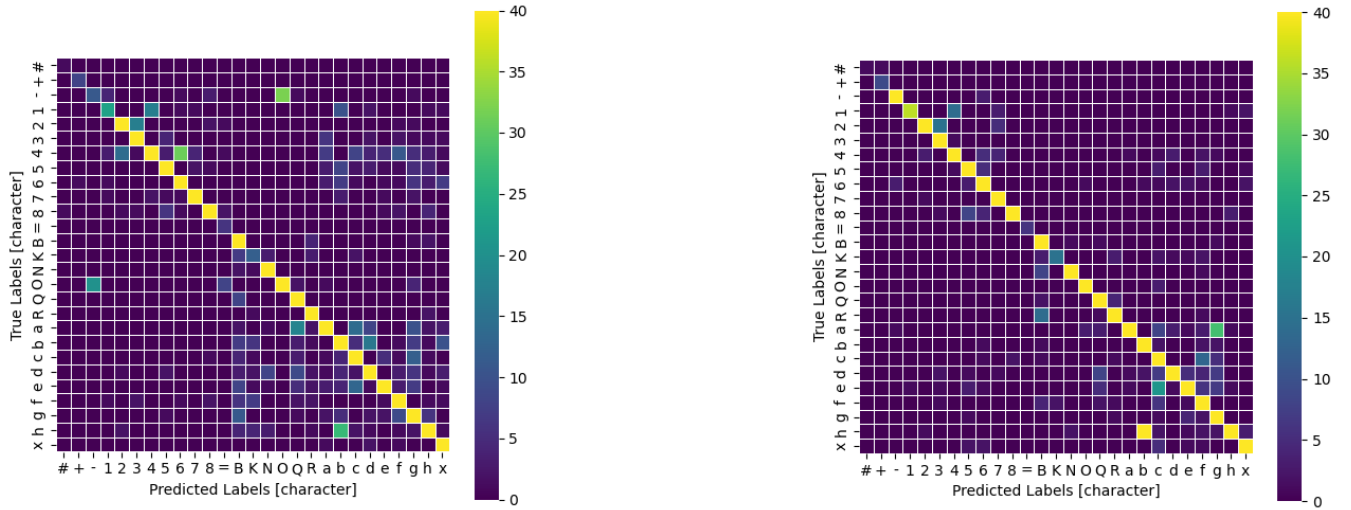


Figure 6: Confusion matrix per character basis for ViT (left) and CNN-BiLSTM (right)

D. Quantitative results: Precision, Recall, F1-score per character basis

Vit					BiLSTM				
Class	Precision	Recall	F1-score	Support	Class	Precision	Recall	F1-score	Support
#	0.50	0.50	0.50	2	#	1.00	0.50	0.67	2
+	0.19	0.80	0.31	10	+	0.75	0.90	0.82	10
-	0.34	0.21	0.26	53	-	0.93	0.94	0.93	53
1	0.55	0.41	0.47	56	1	0.97	0.64	0.77	56
2	0.70	0.68	0.69	79	2	0.91	0.73	0.81	79
3	0.83	0.90	0.87	190	3	0.90	0.98	0.94	190
4	0.82	0.70	0.75	332	4	0.93	0.93	0.93	332
5	0.76	0.85	0.80	221	5	0.89	0.91	0.90	221
6	0.73	0.86	0.79	214	6	0.87	0.91	0.89	214
7	0.88	0.90	0.89	112	7	0.87	0.98	0.92	112
8	0.83	0.78	0.81	93	8	0.94	0.80	0.86	93
=	0.27	1.00	0.43	6	=	1.00	1.00	1.00	6
B	0.72	0.96	0.82	173	B	0.84	0.98	0.90	173
K	0.34	0.44	0.39	27	K	0.88	0.56	0.68	27
N	0.94	0.99	0.97	295	N	0.99	0.97	0.98	295
O	0.51	0.60	0.55	93	O	0.93	0.94	0.93	93
Q	0.79	0.96	0.87	197	Q	0.91	0.97	0.94	197
R	0.84	0.97	0.90	99	R	0.89	0.84	0.86	99
a	0.78	0.64	0.70	167	a	0.96	0.69	0.81	167
b	0.75	0.80	0.77	286	b	0.86	0.97	0.91	286
c	0.79	0.85	0.82	216	c	0.76	0.89	0.82	216
d	0.82	0.80	0.81	240	d	0.95	0.84	0.89	240
e	0.87	0.70	0.78	165	e	0.87	0.70	0.77	165
f	0.74	0.80	0.77	132	f	0.73	0.87	0.79	132
g	0.50	0.65	0.57	113	g	0.66	0.91	0.76	113
h	0.62	0.52	0.57	101	h	0.88	0.45	0.59	101
x	0.87	0.98	0.92	201	x	0.96	0.95	0.95	201

Table IV: Precision, recall and f1-score metric per character of the two best performing ViT and BiLSTM models;

E. Samples of preprocessed data for training and inference of ViT



Figure 7: Overview of the ViT DataLoader: Row 1 corresponds to the training dataset, Row 2 to the validation dataset, and Row 3 to the test dataset.

F. Prediction examples

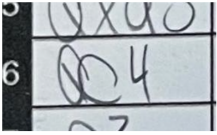
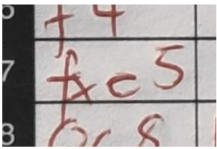
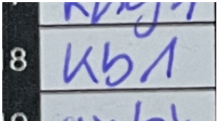
	Image	Label	Prediction	Confidence	CER	WER
1		Qc4	Qf4	0.549	0.3333	1
2		fxe5	Kc5	0.4542	0.75	1
3		Kb1	Kb4	0.9129	0.3333	1

Figure 8: Example predictions from CNN-BiLSTM on test set images, including the actual label, confidence level, CER, and WER.

G. ViT Attention Overlay

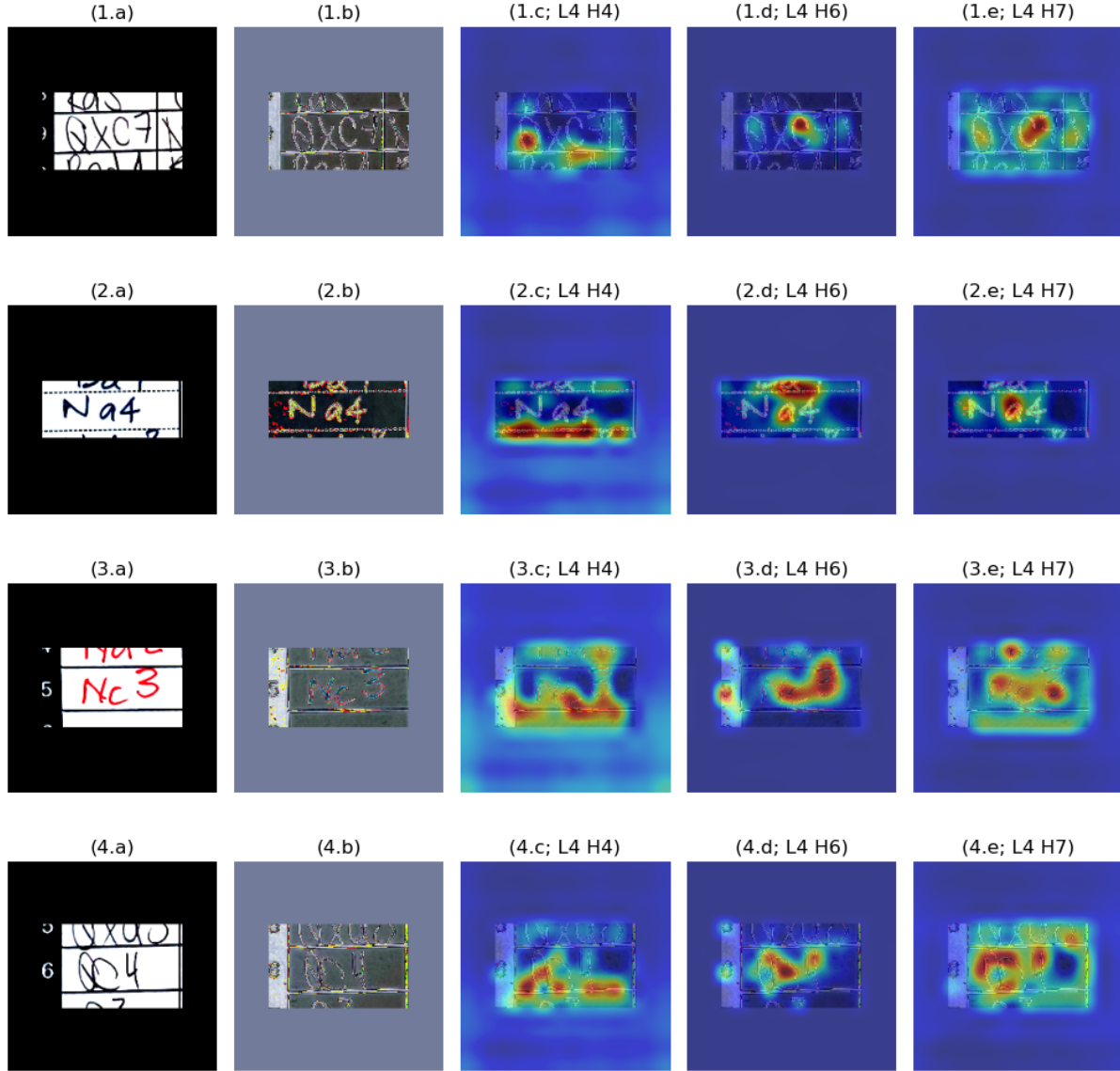


Figure 9: Attention Overlay; Figure (a) represents a raw image, (b) normalised input of the transformer, and (c-e) attention overlay. The attention overlay presents the original input image with the ViT's attention map superimposed as a heatmap (L-layer, H-Head). The overlay highlights the specific regions of the image that the model focuses on when making predictions.

H. Learning rate tuning

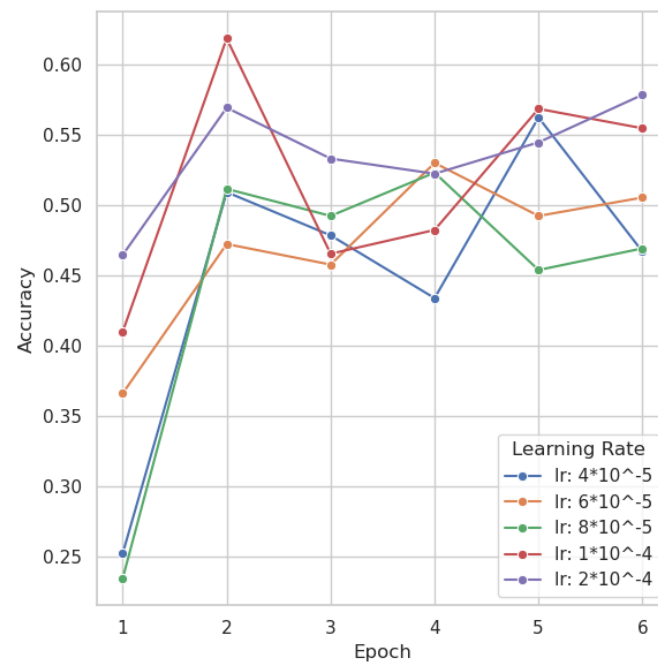


Figure 10: Accuracy obtained on the test set for the ViT model across training epochs with different learning rate values.