**Verification of Handrawn Signature Using Convolutional Networks**

**Project Report**

**Document**

**Course Code:** CS-4059

**Instructor:** M. Burhan Khan

**Group Members**

* Saman Khan 19K-0354
* Muhammad Farjad 19K-0357
* Muhammad Areeb 19K-0360

**ABSTRACT**

Every individual has a signature that is used mostly for personal identification and verification of significant papers or legal transactions. Static and dynamic signature verification are both possible. Static (offline) verification is the process of confirming an electronic or document signature after it has been created, whereas dynamic(on-line) verification occurs as an individual creates his/her signature on a digital tablet or similar device. Due to their simplicity and uniqueness, offline signatures are the most extensively used biometric authentication mechanism in banking systems, administrative and financial applications. Several automated algorithms to predict the authenticity of an offline signature have been developed. However, just a few review articles summarize the existing knowledge on offline signature verification systems based on deep learning. The purpose of this paper is to offer cutting-edge deep learning-based models for signature verification systems employing convolutional neural networks.

**Literature Review**

Offline signatures are used as confirmation of identity on bank checks, loans, properties, and other official documents that are not digital. It is a biometric measure. Biometric refers to the specific information about someone’s body such as a pattern of color in the eyes, handwriting recognition and many more. Offline signature verification is an important process. Offline signatures are typically checked by analyzing the signature's pattern fluency or by visually comparing the signature's pattern to the previously gathered samples. To detect a signature fraud, however, manual offline verification of a large number of documents is laborious and requires human attentiveness, experience, and competence. Based on the picture acquisition method, the verification procedure of offline signatures can be performed either by online or offline. It's worth noting that the online approach is also known as a dynamic signature verification method, whereas the offline approach is known as a static signature verification method. Tablets, pressure-sensitive pens, and other gadgets are all used to collect digital signatures online. Instances where a series of time periods is used to collect the inherent dynamic data of an offline signature. The angle, pressure, and location of the writer's hand are all part of this live data. When acquiring an offline signature, though, a scan is all that's needed (i.e., converting into a digital image).

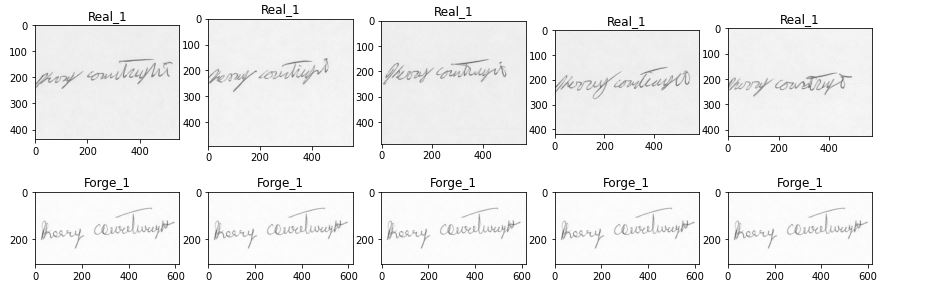
Thus, developments in technology have led to more complex machine learning-based signature verification techniques. Because of this dramatic increase in forgery instances, financial institutions have turned to manual verification of offline signatures. Furthermore, when there is a large volume of bank checks that must be verified by hand, the process becomes even more arduous. As a result of the difficulty of this work, there is an urgent requirement for a reliable computerized offline signature verification system that can automatically tell the difference between authentic and faked offline signatures. The machine learning-based offline signature verification research is widely being made for human verification in access control, employment documentation concerns, identity in finance and security of various applications in recent years. Every machine learning-based offline signature verification system (henceforth referred to as offline signature verification system) has the same overarching goal: to automatically discern between a legitimate and faked offline signature. The offline signature verification system is different from the online signature verification system in the sense that it is not using any intrinsic dynamic information to identify the falsified signature. This inherent dynamic information can only be available in online signature verification system systems. This is why recovering offline signatures from scanned signature images is such a challenge for offline signature verification systems: the system is more complex. It needs to be mentioned that the online signature verification mechanism is out of the scope of this study. Offline signature forgeries can be categorized into three categories: elementary, intermediate, and advanced. In the case of simple forgery, the individual attempting to counterfeit the offline signature has knowledge of the signer's true identity but is unaware of the authenticity of the signer's true signature. However, in random forgeries, either the signer's name or the signer's actual signature is known to the forger but the forger uses the forger's own signature instead. In sophisticated forgeries, both the signer's name and the signer's actual signature are known to the forger, and the forger has likely practiced imitating the signatures in question [4, 6]. It is possible to use a writer-dependent, writer-independent, or mixed offline signature verification mechanism. In terms of categorization accuracy, the most popular method, known as the writer-dependent (WD), has proven to be superior. Each user's verification model is custom-trained. Because offline signature templates are not stored for verification in the WD offline signature verification system [3], the system is secure. When more users are added, however, the complexity and computational cost of this method rises since individual classifiers must be built for each user [3]. The WI method, which requires only a single global classifier for all users, is more user-friendly and efficient. The convenience of only needing to provide a single signature sample to use a WI system has made it more popular than the WD method [3]. Recently, deep learning-based WI Offline signature verification system systems were investigated by Engin et al. [2]. Through testing their proposed stamp cleaning technique on actual papers, they were able to successfully boost offline signature verification's efficiency. Alternately, a hybrid WD-WI Offline signature verification system system can be created by combining the two methods. Offline signature verification systems, as well as online signature verification systems, have been thoroughly examined by Diaz et al. [1], who have also provided insightful commentary on the state of the art in this urgent topic. The most popular offline signature datasets have been analyzed in this overview, along with their differences, preparation methods, characteristics, and verifiers. The performance of several Offline signature verification system systems was analyzed using performance metrics, and this literature review also discussed some of the most pressing topics in the fields of study at the moment. However, it has missed the number of current research domains issues, like the need for a large public dataset for Offline signature verification system system, the role of transfer learning in improving the accuracy of Offline signature verification system systems, cursive handwritten scripts and the usefulness of ensemble classifiers for more accurate results. The fifteen signature verification methods in the literature were also examined by Shah et al. [5], who also categorized them according to machine learning (ML) models and feature extraction approaches. In addition, they summed up the benefits and drawbacks of Offline signature verification system systems. However, the preparation techniques and publicly available datasets utilized in offline signature verification systems are not included in this review. There is also no comparison of the systems' accuracy rates for offline signature verification. The purpose of this system is to demonstrate a trustworthy offline signature verification system that can be used with different datasets, such as CEDAR and Kaggle. These datasets will be compared based on the accuracy produced on both the train set and test set.

**Methodology**

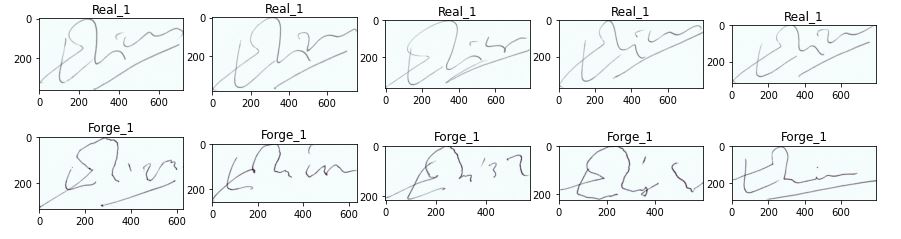
1. **Data Collection**

The data needed for this project is sourced via Kaggle which contains both genuine and fraud signatures of Dutch users. There are a total of 2,133 images belonging to two classes. Other than this we have also used cedar dataset which also contains both genuine and fraud signatures. Each of 55 individuals contributed 24 signatures thereby creating 1,320 genuine signatures. Some were asked to forge three other writers’ signatures, eight times per subject, thus creating 1,320 forgeries.

* **CEDAR Dataset**

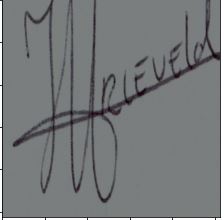
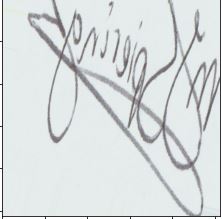
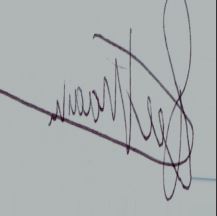
****

* **KAGGLE Dataset**

****

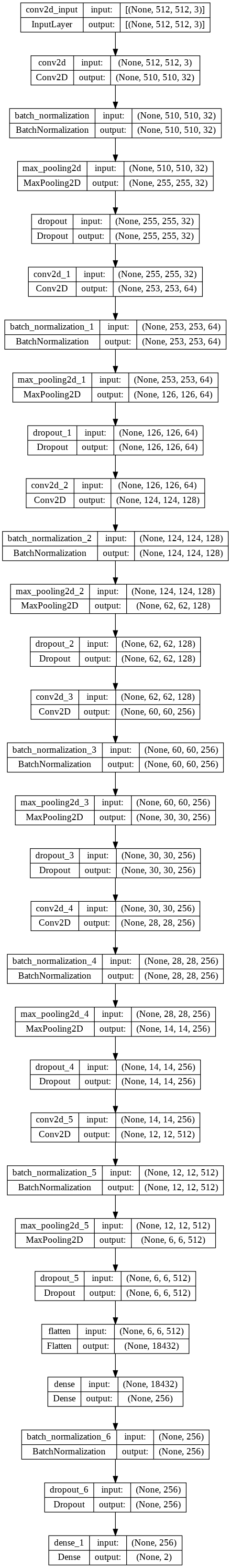
1. **Data Preparation**

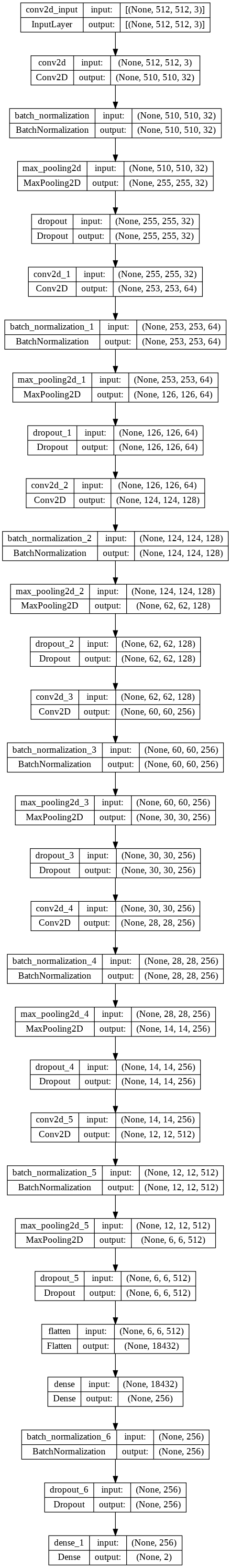
The data obtained from Kaggle was already split into training and testing sets, however the data was suited to writer dependent signatures as each individual had its own separate folder. To overcome this, we moved all the genuine signatures into a folder named real and moved all the forged signatures into a folder named forged. We repeated the same step for both training and testing. As for the cedar dataset it was already divided into real and forged categories. After this the data was fed into the Keras Image Generator function where operations such as rotations, rescaling, and horizontal flips are carried out randomly. The data is processed in batches of of size 32, where shape is set to (512, 512).

**  **

1. **Model Training**

For this purpose, we employ convolutional neural networks (CNNs, or ConvNets), a type of artificial neural network that has been effectively used to analyze visual data. Inspiringly, the CNN's interconnection network between neurons is similar to the structure of the visual cortex in animals. When it comes to implementing CNN in our work, we rely on the Keras package and its TensorFlow backend. After loading the directory of processed photos, we train the model and assess its efficacy. To facilitate the use of the Keras Python library, the signature images in this work are organized in a specific file directory structure. Then, the convolutional neural network (CNN) has been built in python using the Keras and TensorFlow backend to learn the characteristic pattern associations. Our model consists of a total of 6 layers. Each layer has four components that are: convolution, batch normalization, max pooling, and dropout. After the picture has undergone the convolution process, the feature map generated by the convolution is normalized by the batch normalization layer, a max pooling layer receives the output from previous layer and uses them to generate pooled feature maps, these pooled features are then passed to dropout layer which drop 25% neurons to prevent the model from overfitting. The resulting pooled feature map is then fed into the subsequent convolution layer, and so on, until the final dropout layer is reached. The final feature map is flattened before being transferred to the fully connected layers which is then further normalized and finally softmax activation function is applied to make the final decision. The accuracy and loss measures are then used to check the model's validity and determine how well it fits the data. It is possible to make a prediction after training the model through numerous iterations of forward and backward propagation.





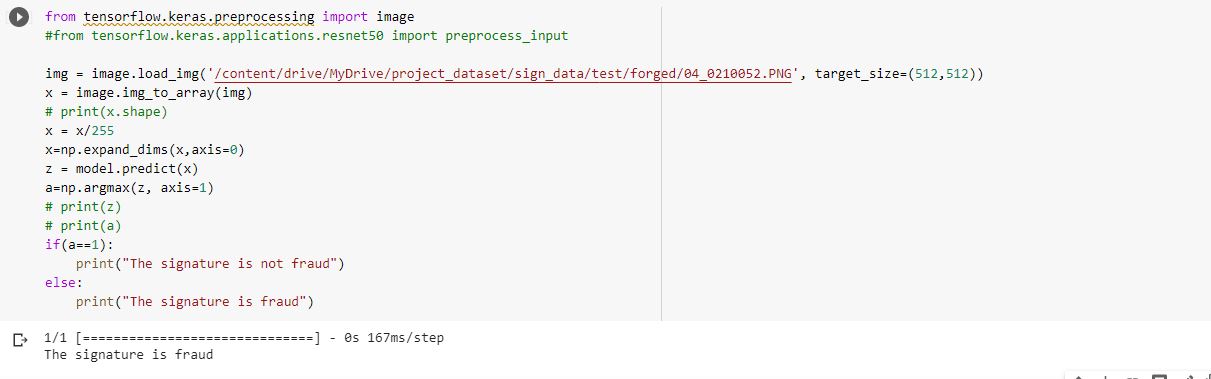
1. **Model Testing**

A signature from a holdout set was used to verify the accuracy of the model's predictions. Convolution, batch normalization, max pooling, and dropout layers are interleaved with one another and applied to an image in a series in our implementation. Following outputs demonstrate prediction of our model on single test image from both cedar’s and kaggle’s dataset.

* **CEDAR**

****

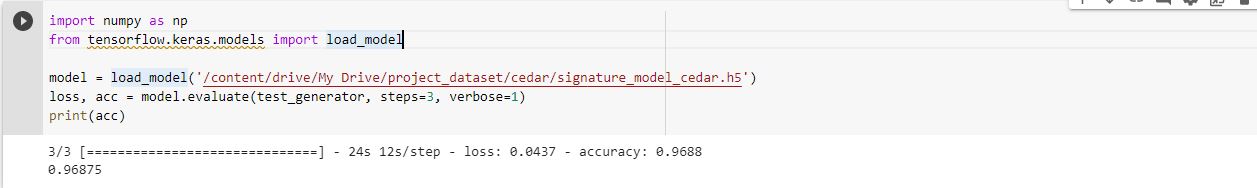
* **KAGGLE**



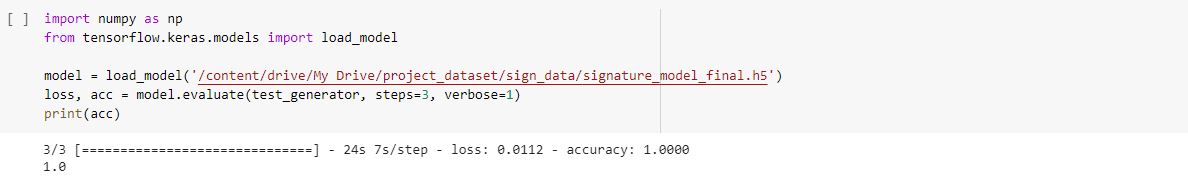
1. **Model Evaluation**

The model was evaluated using accuracy score. The accuracy score of training and testing set of both datasets were compared to observe model’s performance.

* **CEDAR’S Accuracy**



* **KAGGLE’S Accuracy**



**Brief Note on Training/Testing**

Five convolutional layers were initially incorporated into the model, and they were interleaved with max pooling layers. While we saw a rapid improvement in training loss and accuracy as we began training, we saw a steady increase in validation loss and a plateau in validation accuracy throughout the validation phase. Since it was obvious that the model was over fitting, we added drop out and batch normalization layers to mitigate the problem. To improve the model, we retrained it and achieved 95% accuracy during training, but only 70% accuracy during validation. Still struggling with the over fitting problem, we gave our model a second opportunity by including a sixth layer of convolution, max pooling, batch normalization, and dropout. This time around, a substantial enhancement to the model was seen. The accuracy in both training and validation appeared to be improving over time. This is the model we've decided to use moving forward. Aside from batch normalization and dropout layers, we also used data augmentation, which helped improve accuracy.

* **CEDAR’S Training Pattern**



* **KAGGLE’S Training Pattern**

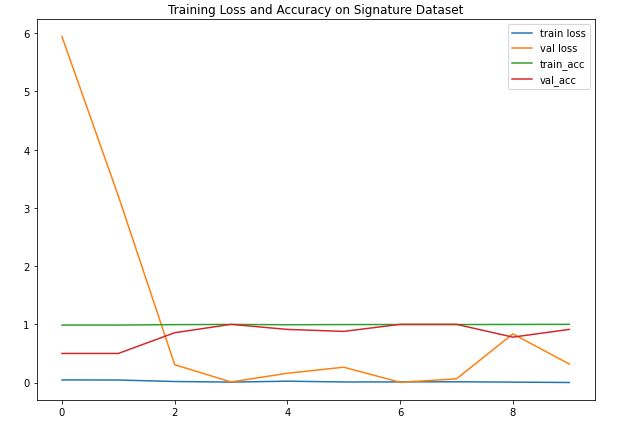
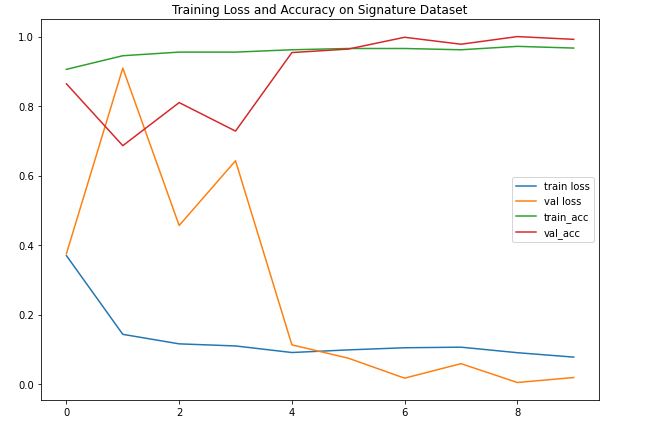


**Results**

We begin by adjusting the image to a standard before continuing with the processing. We next perform the necessary file management and manipulation tasks to divide the image batches according to the specified split ratio. We build models using a variety of pairs of data and compare their training and validation accuracy using a line plot to identify overfitting and underfitting. In the accompanying table and graph, we detail the accuracy and loss experienced across the various datasets.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Dataset** | **Train Accuracy** | **Train Loss** | **Test Accuracy** | **Test Loss** |
| Kaggle | 100% | 1.35% | 100% | 1.12% |
| Cedar | 95.83 | 8.69% | 96.88% | 4.37% |

**CEDAR KAGGLE**

****

**Conclusion/Future Directions**

It has been demonstrated that it is possible to implement a model that can learn from signatures and make predictions about whether or not a given signature is a counterfeit. Forgery detection is trained on the entire image collection and every time the calculations are performed, it reduces the risk of error in classification, and the popularity pattern is learned using Convolutional Neural Networks, which perform well with the dataset of more than 2,000 photographs. The CNNs are used to learn the signatures where each user is represented by two classes in the resulting model. A 100% accuracy on Kaggle dataste and 96.88% on Cedar dataset was the most we could achieve. There are various uses for a trustworthy and accurate signature recognition and verification system, including law enforcement, security management, and numerous commercial procedures. It serves as a middleman for the verification of various papers (checks, certificates, legal documents, etc.). Because signatures are so commonly used for personal authentication on diverse financial documents like bank checks, this approach contributes to our Sustainable Development Goal (SDG) of Industry Innovation and Infrastructure. Signature authentication reveals so much data, signature fraud can have devastating effects. This necessitates the validation of signatures on financial papers to protect the confidentiality of information. Our cutting-edge solution will automate and simplify authentication for bank staff, saving everyone time and effort. The proposed system is very cost-effective in detecting and investigating forgeries in real-time, and its duty may be increased by training the extracted features on the Artificial Neural Networks by storing the extracted features. The accuracy of the resulting system could be improved in future study by experimenting with other parameter coefficients to increase the gap between authentic and counterfeit signatures and improving the system's implementation so that it can process digital signatures as accurately as traditional ones.

**References**

[1] Diaz, M., Ferrer, M. A., Impedovo, D., Malik, M. I., Pirlo, G., and Plamondon, R. (2019). A perspective analysis of handwritten signature technology. ACM Computing Surveys (CSUR), 51(6):117.

[2] Engin, D., Kantarci, A., Arslan, S., and Kemel Ekenel, H. (2020). Offline signature verification on real-world documents. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, pages 808–809.

[3] Eskander, G. S., Sabourin, R., and Granger, E. (2013). Hybrid writer-independent–writer-dependent offline signature verification system. IET biometrics, 2(4):169–181.

[4] Hafemann, L. G., Sabourin, R., and Oliveira, L. S. (2016). Writer-independent feature learning for offline signature verification using deep convolutional neural networks. In 2016 International Joint Conference on Neural Networks (IJCNN), pages 2576–2583. IEEE.

[5] Shah, A. S., Khan, M., and Shah, A. (2015). An appraisal of off-line signature verification techniques. International Journal of Modern Education and Computer Science, 7(4):67.

[6] Zois, E. N., Alexandridis, A., and Economou, G. (2019). Writer independent offline signature verification based on asymmetric pixel relations and unrelated training-testing datasets. Expert Systems with Applications, 125:14–32.