

Image Classification Using Convolutional Neural Networks for Automatic Correction of Multiprova Answer Sheets

Classificação de Imagens Utilizando Redes Neurais Convolucionais para Correção Automática de Cartões Respostas do Multiprova

Darlan de Castro¹, Andre Bessa², Rex Medeiros², Helton Maia²

Abstract: During the COVID-19 pandemic, the Federal University of Rio Grande do Norte (UFRN) adopted the emergency remote regime for its academic activities. In this scenario, the Multiprova software played a key role in facilitating the teaching-learning process at the institution. With the resumption of face-to-face classes, the need to improve the evaluation of tests arose. One of the solutions implemented was the automatic correction of answer cards filled in by hand by students. The main focus of this project is to optimize the proof correction procedure in the Multiprova software by applying handwritten character recognition techniques through convolutional neural networks. Answer-card images, obtained directly from the students, were thoroughly analyzed to develop a neural classifier employing convolutional neural networks. The results obtained revealed remarkable accuracy, surpassing 98%. This implementation has provided significant advances in performance and reliability in correcting tests, strengthening the effectiveness of the software.

Keywords: COVID-19, Multiprova, Handwritten Character Recognition, Convolutional Neural Networks.

Resumo:

Durante o período da pandemia de COVID-19, a Universidade Federal do Rio Grande do Norte (UFRN) adotou o regime remoto emergencial para suas atividades acadêmicas. Nesse cenário, o software Multiprova desempenhou um papel fundamental ao facilitar o processo de ensino-aprendizagem na instituição. Com a retomada das aulas presenciais, surgiu a necessidade de aprimorar a avaliação de provas. Uma das soluções implementadas foi a introdução da correção automática de cartões-resposta preenchidos à mão pelos alunos. O foco principal deste projeto é otimizar o procedimento de correção de provas no software Multiprova, por meio da aplicação de técnicas de reconhecimento de caracteres manuscritos através de redes neurais convolucionais. Imagens de cartões-resposta, obtidas diretamente dos estudantes, foram minuciosamente analisadas para desenvolver um classificador neural utilizando redes neurais convolucionais. Os resultados obtidos revelaram uma notável precisão, ultrapassando os 98%. Essa implementação tem proporcionado avanços significativos em desempenho e confiabilidade na tarefa de correção de provas, fortalecendo a eficácia do software.

Palavras-Chave: COVID-19, Multiprova, Reconhecimento de Caracteres Manuscritos, Redes Neurais Convolucionais.

¹ Department of Computer Engineering and Automation, UFRN, Brazil

² School of Science and Technology, UFRN, Brazil

*Corresponding author: helton.maia@ufrn.br

DOI: <http://dx.doi.org/10.22456/2175-2745.130645> • Received: 06/03/2023 • Accepted: 03/11/2023

CC BY-NC-ND 4.0 - This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.

1. Introduction

John McCarthy, a founding father of artificial intelligence (AI), defined it as "the science and engineering of making intelligent machines, especially intelligent computer programs. It is related to using computers to understand human intelligence, but AI does not have to confine itself to biologically observable methods" [1]. This definition presents AI as a field of study focused on creating algorithms and systems that can

perform tasks that require human-level intelligence, such as perception, reasoning, and learning. With Artificial Intelligence, it is possible to gain productivity, process optimization, and more reliable complex data analysis.

As explored in *Artificial Intelligence: A Modern Approach* [2] AI is divided into several areas. Among them, we can mention Machine Learning (ML), which focuses on developing algorithms and statistical models that allow computers to improve their performance in a specific task through experience.

Machine Learning aims to create algorithms that can learn from data, identify patterns, and make predictions or decisions without being explicitly programmed. This differs from what programmers do in traditional programming when the data is inserted in a ready-made algorithm. In other words, algorithms are created to learn from data rather than being explicitly programmed to perform a particular task.

Deep learning is a subfield of machine learning that focuses on using artificial neural networks with many layers to model and solve complex problems. These models are trained on large amounts of data. They can automatically learn hierarchical representations of the data, allowing them to perform tasks such as image classification as the *Artificial intelligence-based image classification methods for diagnosis of skin cancer: Challenges and opportunities* [3], speech recognition as seen in *Speech emotion classification using machine learning algorithms* [4], techniques for object recognition [5], and natural language processing with high accuracy as in *Evaluation of machine learning methods for natural language processing tasks* [6]. The key innovation of deep learning is using deep neural networks, which can automatically learn multiple levels of abstraction from the data, making it the proper choice for many applications.

A Convolutional Neural Network (CNN) is a deep learning neural network created primarily for image recognition and processing. It is a variant of the traditional neural network, where the structure and weights of the model are learned from training data. CNNs use a process called convolution to extract features from images and reduce the dimensionality of the data. The convolution operation involves passing a filter over the image and computing dot products between the filter and the image. The process is repeated several times using a different filter to extract multiple features from the image. The output from the convolution layers is then fed into fully connected layers, where the features are combined to make a final prediction. This process allows CNNs to capture an image's local and global features, making them suitable for a wide range of computer vision tasks.

This work was initially motivated by the need to improve the educational software called Multiprova [7] for automating the correction of handwritten answer sheets. Multiprova was created as a student assessment system and gained an online module due to the Covid-19 pandemic. With the return to face-to-face classes, the system had to adapt to the written tests, and for this, an answer sheet was created for multiple-choice, matching, and true or false questions. From this problem, the resolution process consists of processing the answer sheet image and isolating the characters present for their classification, developing a character recognition process that makes the correction of the quick test efficient. Image processing techniques and convolutional neural networks were essential for the development of this research.

2. Methods

2.1 Computational development

All computational development, including programming languages, tools and libraries used in this work will be presented in this session.

2.1.1 Programming languages

In this paper, two programming languages were used. Python was used for image processing of answer sheets and creating and training the neural network. The second language, Javascript, was used to port the trained network's settings to the mobile environment to predict students' answers on the sheets.

2.1.2 Libraries and auxiliary software

Created and maintained by Google, TensorFlow [8] is an open-source library for numerical computing and Machine Learning. This library groups several Deep Learning algorithms and models, simplifying their use for the end-user. Due to its ease of use and integration with Python, TensorFlow is currently the most widely used tool for Machine Learning development.

Built on top of TensorFlow, Keras [9] is a high-level API for creating and training Deep Learning models. With easy-to-understand and apply elements, this API is used in TensorFlow's Machine Learning projects as it is present within its ecosystem and can be imported directly from the TensorFlow package into the code.

OpenCV, or Open Source Computer Vision [10], is an open-source library initially developed by Intel in 2000 to facilitate the development of projects focused on computer vision. It is implemented in several programming languages and is primarily used for analyzing and manipulating images and videos. It is the leading library for computer vision as it contains a good enough abstraction of functions and algorithms that would be complex to implement manually.

React Native is a JavaScript library created by Facebook to create and maintain native mobile applications (Android and IOS). The great advantage of using a library like this is the ease of learning one language (JavaScript) and natively developing for two different platforms - Android, which uses Java, and IOS, which uses Objective-C and Swift.

Weights and Biases (WandB) [11] is a platform for managing and visualizing machine learning experiments. It provides tools for tracking and visualizing the performance of models during training, making it easier for developers to optimize their models and improve their results. The platform offers many features, including real-time dashboards, performance logging, model versioning, and automatic reports.

2.1.3 Automatic sheets correction

Initially, the answer sheet image must undergo an initial treatment to obtain a standard format. The purpose of this treatment is to normalize the image's perspective, making it appear as if it was captured parallel to the camera. Additionally, its rotation must be adjusted, which can be achieved using the position references on the sheet. The sheet can be visualized in Figure 1.

Figure 1. Answer sheet used by students.

Once the formatted sheet image is obtained, predefined cuts are made in all 110 text boxes. In this way, 11 correspond to the student's enrollment number, 3 to the test code, and 96 are evenly distributed among the 16 questions (6 for each). These images are then extracted and grouped for further analysis.

Considering that the main challenge of this problem is the recognition of handwritten characters, the most appropriate approach is the use of Machine Learning, precisely the recognition of images via artificial neural networks. Thus, the development of CNNs capable of identifying letters and numbers accurately is crucial to this problem.

2.2 Dataset

Before training CNNs, a large and diverse amount of data must be collected, specifically handwritten samples of uppercase letters (A-J and T) and single-digit numbers (0-9).

To gather these samples, students from the School of Science and Technology (ECT) at UFRN were asked to complete a sheet with examples of the required characters. Although each student produced an equal number of characters for each class, the number of images per class was not uniform due to a filtering process that discarded low-quality images, resulting in some classes having more samples than others.

To improve neural network training, it is necessary to balance the class sample sizes for all datasets (letters, digits, and true/false). Figures 2, 3, and 4 show the initial imbalance of the classes for the created datasets.

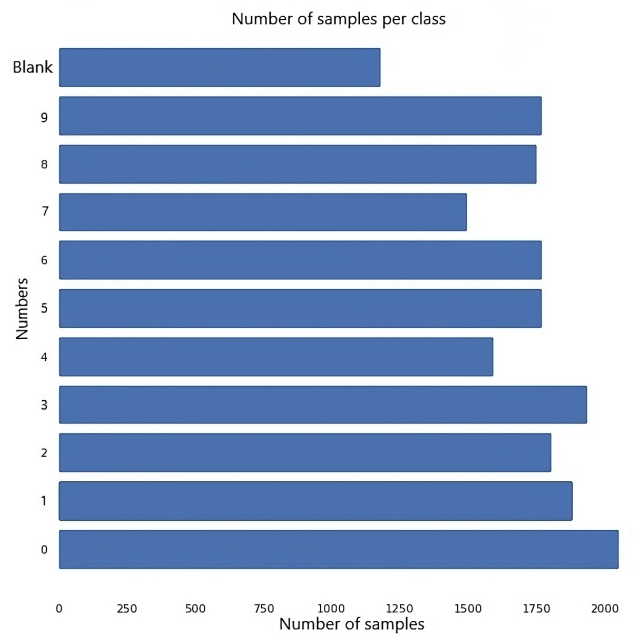


Figure 2. Number of samples per class (digits).

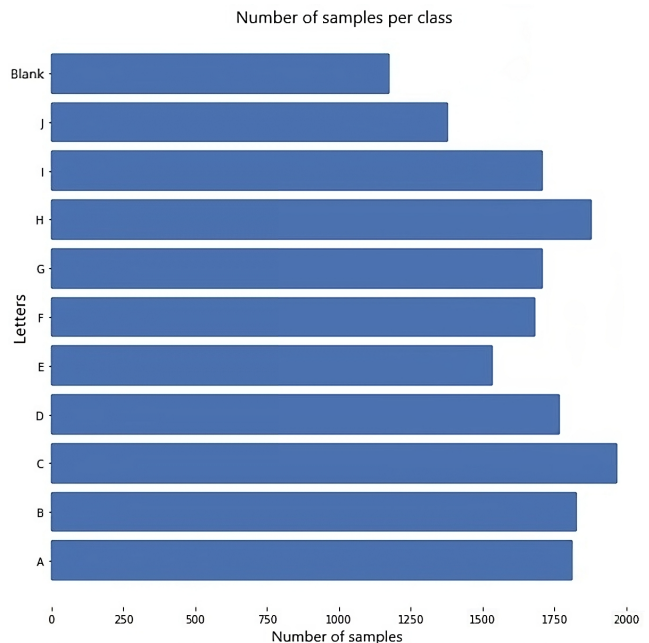


Figure 3. Number of samples per class (letters).

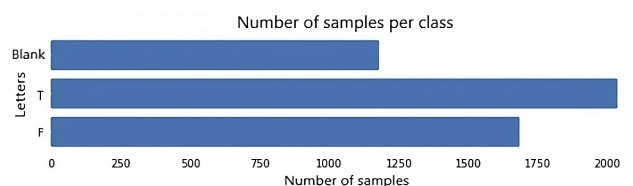


Figure 4. Number of samples per class (true/false).

Applying data balancing techniques is recommended to achieve a healthy distribution of samples per class and im-

prove the dataset quality. Oversampling is a technique that can be used for this purpose.

Oversampling is effective when a class has a significantly lower number of samples than the others. To equalize the number of samples, one should select images from the under-represented class and modify them to generate new images that still belong to that class. To preserve the original characteristics of the images, the oversampling techniques used in this work were zoom in or out, translation, and rotation of the images.

2.3 Image processing techniques

In order to address the discrepancy in sample numbers, pre-processing is necessary. One effective method for increasing the diversity of data in the dataset is to generate new images by applying mathematical transformations to the base samples. The images, in this case, have dimensions of 32 x 32 x 3 (height x width x number of color channels). Figure 5 provides an example of an image sample from the G class.

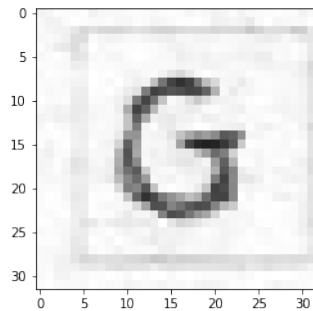


Figure 5. Sample of G class

Figures 6 display the results of the data augmentation process, comprising nine images for each class. Comparing these new samples with the original shown in Figure 5 reveals visible effects of zoom, rotation, and displacement.

In the letters dataset, as shown in Figure 3, the number of samples per class varies, with the Blank class containing approximately 1100 samples and the C class having 2000 samples. This disparity represents an 81% difference in sample numbers between these two classes. Similarly, in the numbers dataset (Figure 2), the Blank class has 1100 samples, while the 0 class contains 2100 samples, resulting in a maximum difference of around 90% in sample numbers between these classes.

These differences in sample numbers within the dataset could lead to training issues and classification errors. The CNN may perform well in identifying classes with a large number of samples but fail to identify those with a low number of samples.

Arbitrary deformations in the rectangular reference area extracted from the original photo produce nonhomogeneous deviations in the location of the segmentation sites. In order to avoid any loss of information, we decided to deal with an auxiliary region obtained after segmentation over a wider region calculated from the original coordinates. Then, for

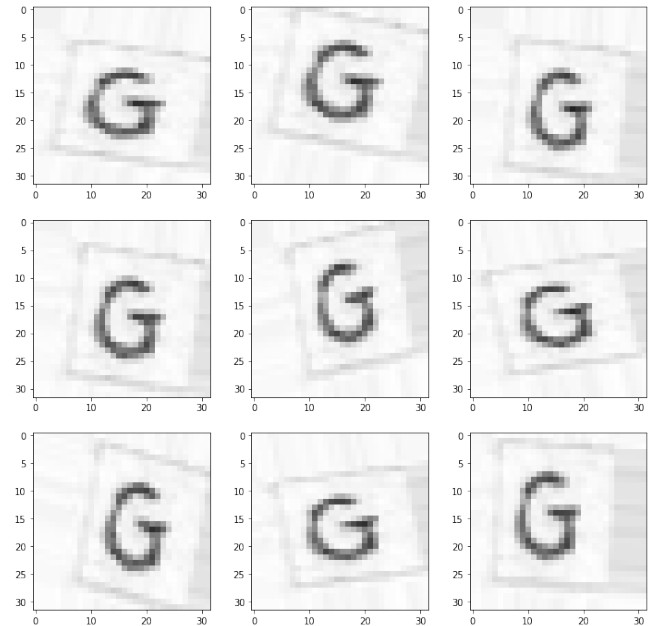


Figure 6. Sample of G class after augmentation

each region, we calculated the centroid of the auxiliary image as a reference point to the final segmentation.

The direct calculation of the centroid does not produce the expected result due to the influence of any reminiscent shadows carrying a large pixel weight. However, a simple thresholding of the auxiliary image using the average intensity as the cutoff parameter was enough to eliminate spurious deviations in our context.

Figures 7 and 8 display the before and after, respectively, of the centroid's calculation.

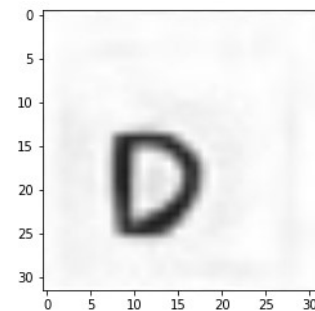


Figure 7. Sample before centroid's calculation

2.4 Convolutional neural networks (CNN)

The developed CNN models use convolutional, maximum pooling, flattened, and fully connected layers in their construction. Basic structures were used in modeling the CNNs, being adapted as empirical tests were performed.

WandB allows empirical tests to be carried out with different hyperparameter configurations in search of the best results for the model.

Base structure for CNNs classifying letters and num-

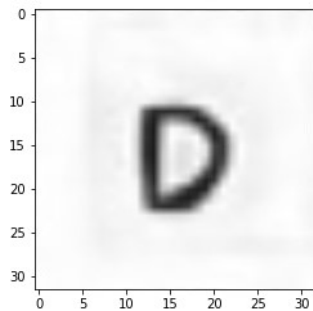


Figure 8. Sample after centroid's calculation

bers: The first layer is a convolutional layer with variable filters and kernel size, which is used to extract features from the input image. The second layer is a maximum pooling layer with a 2×2 filter, which is used to downsample the feature maps obtained from the previous layer. Layers three, four, five, and six repeat the pattern above (convolutional followed by maximum pooling). The seventh layer is a convolutional layer with variable filters and a fixed 2×2 kernel size. The eighth layer is a flatten layer, which takes the previous layer's output and flattens it into a one-dimensional array. The ninth and final layer is a fully connected layer with variable filters and an output of 11 neurons.

The properties modified for the letter and number models are displayed in Table 1:

Table 1. Hyperparameter for CNN specialized in Letters and Numbers

Hyperparameter	Values
Learning rate	0.001 to 0.01
Training epochs	20 to 50
Convolutional kernel size	2, 3
Filters	32 to 64

Base structure for CNN true (T) or false (F) classifier:

The first layer is a convolutional layer with variable filters and kernel size. The second layer is a maximum pooling layer with a 2×2 filter. Layers three, four, five, and six repeat the pattern above (convolutional followed by maximum pooling). The seventh layer is a flatten layer, which takes the previous layer's output and flattens it into a one-dimensional array. The eighth and final layer is a fully connected layer with variable filters and an output of 3 neurons.

The properties modified for the true or false model are presented in Table 2:

Table 2. Hyperparameter for CNN specializing in True or False.

Hyperparameter	Values
Learning rate	0.001 to 0.01
Training epochs	2 to 10
Convolutional kernel size	2, 3
Filters	32 to 64

The reduced epochs in the T or F CNN are due to the more straightforward task of classifying three classes instead of the 11 in the letter and number CNNs. Even this low number of epochs already guarantees high accuracy rates.

The optimizer used was Adam (or Adaptive Moment Estimation Algorithm), proposed in 2014 by Diederik P. Kingma and Jimmy Ba. According to its creators, this optimizer is computationally efficient, requires little memory, is invariant to diagonal gradient rescaling, and is suitable for problems that involve large amounts of data or parameters [12].

3. Results and Discussion

In general, the CNN training occurred adequately, allowing for results with accuracy above 98%. Variations of hyperparameters during training proved helpful since a fine adjustment of values can be performed in search of more significant precision in training. Consequently, the objective is also to maximize the accuracy of the validation data set. All the CNNs were trained 20 times with different hyperparameters each time.

3.1 Letters CNN

In Figure 9, one can observe the accuracy metric evolution curves over the validation dataset for the top five trainings. The graph shows curves with similar developments. With the rapid growth in the initial epochs, the curves stabilize and maintain slow development until reaching close to 0.98 at the end of their training.

The curves also maintain similar behavior for the error lines in the validation dataset - Figure 10. With a more accelerated convergence during the first ten epochs and considerably slower afterward, the loss curves approach the value of 0.1.

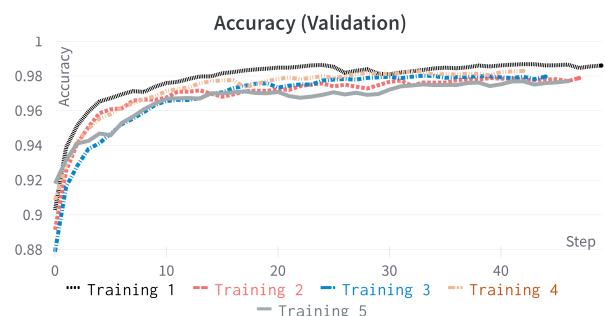


Figure 9. Validation accuracy in 5 trainings (letters)

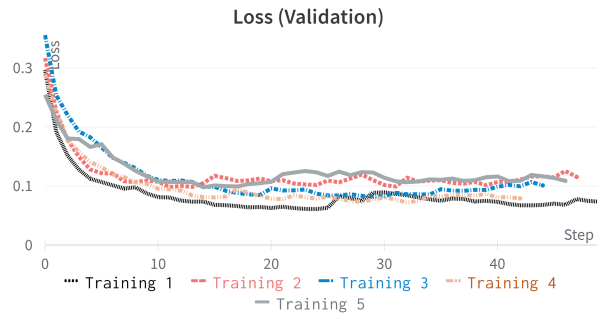


Figure 10. Validation loss in 5 trainings (letters)

The accuracy of the best training in the letter's CNN was 98.6%.

3.2 Numbers CNN

Figure 11 displays the accuracy metric evolution curves for the top five trainings over the validation dataset. The curves exhibit similar trends, with rapid growth in the initial epochs, followed by stabilization and slow development, eventually reaching close to 0.98 by the end of the respective trainings.

Similarly, Figure 12 portrays the error lines in the validation dataset, and the curves show a comparable pattern. The loss curves converge more rapidly during the first ten epochs and slower after that, ultimately approaching the value of 0.11.

Both the accuracy and error curves converge to a value closer to each other, indicating greater consistency of CNN training even with varying hyperparameters.

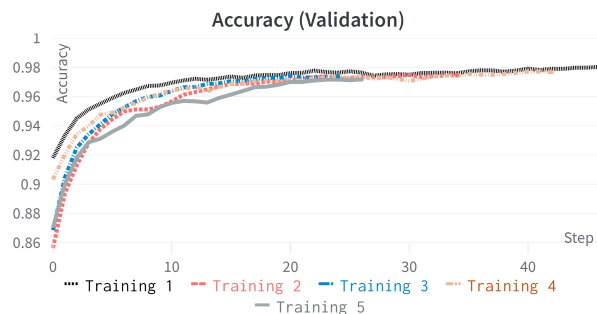


Figure 11. Validation accuracy in 5 trainings (numbers)

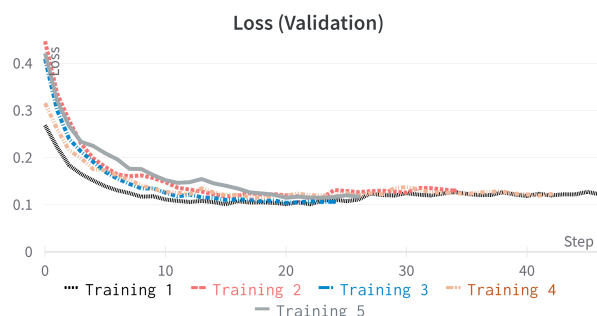


Figure 12. Validation loss in 5 trainings (numbers)

The best training in the number's CNN reached an accuracy of 98.3%.

3.3 True or False CNN

Figure 13 displays the accuracy metric evolution curves for the top five training sessions over the validation dataset, indicating similar trends of nearly linear growth and reaching accuracy above 0.99.

Similarly, Figure 14 depicts the error lines for the validation dataset, displaying a comparable pattern with loss curves that converge towards 0.01.

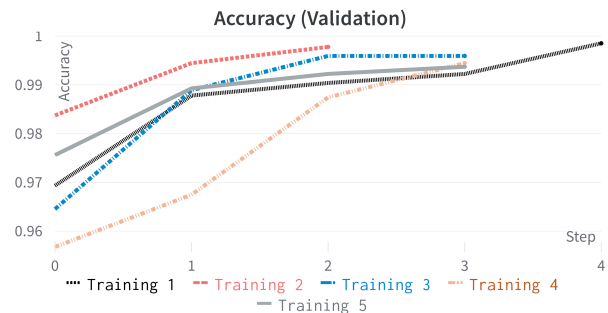


Figure 13. Validation accuracy in 5 trainings (true or false)

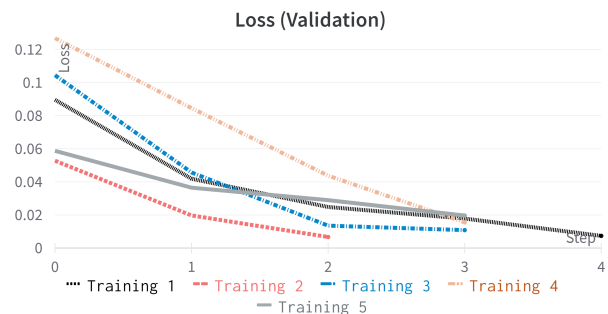


Figure 14. Validation loss in 5 trainings (true or false)

The true or false CNN reached an accuracy of 99.85% in its best training.

3.4 Hyperparameters and confusion matrix

The confusion matrix shown in Figure 15 of appendix A confirms the results presented by the accuracy in Figure 9 and the error in Figure 10 for CNN specialists in letter classification. With the relationships between the 11 classes displayed, one can verify the matrix's main diagonal, with all cells almost filled by the black bar, indicating a good accuracy rate. The cells outside the main diagonal that are not filled also confirm the good results with low error rates.

Figure 16 of appendix A highlights the hyperparameters chosen among the top five trainings. Some convergences between the values of these parameters are evident. The number of epochs, filter 1, filter 2, and learning rate presented similar values for the analyzed training, indicating that these

measures significantly impact the CNNs good accuracy more than the others.

For the number's CNN, the high accuracy rate observed in Figure 11 and the low error shown in Figure 12 can be confirmed in the confusion matrix shown in Figure 17 of appendix B, which represents the relationship between the classes of this CNN. The matrix exhibits the relationships between the 11 classes, and the black bar almost fills the main diagonal cells, indicating high accuracy. The cells outside the main diagonal that is not filled also demonstrate low error rates.

Figure 18 of appendix B highlights the hyperparameters selected among the top five trainings. The values of these parameters converge significantly. Filter 3, filter 4, kernel size, and learning rate exhibit comparable values for the analyzed training, implying that these measures have a more substantial impact than others on CNNs accuracy.

The high accuracy rate observed in Figure 13 and the low error shown in Figure 14 can be confirmed in the confusion matrix of the True or False CNN, shown in Figure 19 of appendix C, which represents the relationship between the three classes of this CNN. The black bar almost fills the main diagonal cells, indicating high accuracy, while the cells outside the main diagonal that are not filled show low error rates.

In Figure 20 of appendix C are highlighted the selected hyperparameters from the top five trainings, revealing significant convergence in their values. The number of epochs, filter 1, dense filter, and learning rate exhibit comparable values, implying that these measures have a more substantial impact on the CNNs accuracy than others.

Besides the statistical validation presented by confusion matrices, it's possible to process the training data to generate a more focused approach to find the more relevant hyperparameters to each CNN. In appendix D it's possible to observe the results of this new metric.

To create these images, the best 5 trainings in each CNN were selected and used by WandB's system of hyperparameter evaluation. The system is designed to classify the hyperparameters based on levels of *importance* and *correlation* given a specific training metric - in this case, the accuracy in the validation dataset was chosen.

Starting with Figure 21, it's possible to observe that the three more important hyperparameters are Learning Rate, Filter 1, and Filter Dense. This means that these parameters should be given special attention during the configuration of the CNN. Filter 2 and Filter 4 have minimal values for both importance and correlation, resulting in small relevance in the CNN training. The only parameter that achieved a high value for correlation ($> |0.7|$) is Learning Rate, helping confirm its relevance to the CNN configuration.

Following Figure 22, the three more essential hyperparameters: Filter Dense, Epochs, and Filter 2, show that these parameters should be configured with caution during the design of the CNN. Filter 4 and Filter 1 have values close to 0 for both importance and correlation, resulting in non-relevance during

the CNN parameter configuration. As the Letters CNN, only one parameter achieved a high value for correlation (Epochs), which helps confirm its relevance to the CNN configuration.

Finally, for Figure 23, with the small number of classes (3) and high accuracy presented by the CNN ($> 99\%$), all the parameters have similar importance values, with none of them having significant emphasis, what shows that the modification of one or more parameters does not change the results in a large scale. The correlation was not displayed in the figure because all the parameters have similar levels of importance, with none having a significant highlight.

In summary, generating such data helped find the key parameters of the trained CNNs. It was discovered that the Filter Dense parameter plays a relevant role in all three CNNs, appearing at least in the top 3 most important parameters in all evaluated hyperparameters.

4. Conclusion

This study aimed to devise a method for recognizing handwritten characters, thereby streamlining the process of grading answer sheets within the Multiprova software. This approach has now been seamlessly integrated and serves as a tool for automatically evaluating tests via smartphones.

Based on the achieved outcomes - a remarkable 98.3% accuracy for the numerical CNN, 98.6% accuracy for the alphabetical CNN, and an impressive 99.85% accuracy for the true-or-false CNN - these figures denote an exceptionally high average success rate. Notably, there is still room for potential enhancements. Using these results, a practical step was taken to export the weight files of the trained CNNs, incorporating them into the Multiprova application (Multiprova Corretor). Developed utilizing React Native technology, this application seamlessly accommodates Tensorflow and its corresponding libraries.

The Multiprova Corretor application is readily accessible through the Play Store virtual marketplace. This empowers educators to download and employ the application on their devices effortlessly, eliminating the need for intermediary resources (such as traditional scoring machines utilized in the earlier version of Multiprova) for automated test grading. To access the application, kindly follow this link: <https://play.google.com/store/apps/details?id=br.ufrn.multiprova>.

Acknowledgements

This work was supported by the School of Sciences and Technology at the Federal University of Rio Grande do Norte (ECT-UFRN).

Author contributions

Author 1 actively participated in the Multiprova project while pursuing a degree in computer engineering at UFRN. Their involvement encompassed tasks such as configuring and training CNNs, image generation, adaptation, and contributing to the paper's composition.

Author 2 assumed a pivotal role as the research group leader dedicated to Multiprova's advancement. Their responsibilities included conceptualizing the answer sheet layout and executing comprehensive image processing before the prediction phase.

Author 3, a vital member of the Multiprova research team, served as a technical leader. Their contributions encompassed shaping software architectures, orchestrating development teams, and executing select programming tasks, notably fine-tuning neural network outputs through post-processing.

Author 4, who specialized in computer vision and artificial intelligence, guided all phases of development and modeling oriented to machine learning. In addition, he supervised the revision of the text, resulting in the final version of this work.

[11] BIEWALD, L. *Experiment Tracking with Weights and Biases*. 2020. Software available from wandb.com. Disponível em: <https://www.wandb.com/>.

[12] KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

References

- [1] MCCARTHY, J. *WHAT IS ARTIFICIAL INTELLIGENCE?* 2007. Online. Disponível em: <http://jmc.stanford.edu/articles/whatisai.html>. Acesso em: 07 jan. 2022.
- [2] RUSSELL, S.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. 3. ed. [S.l.]: Prentice Hall, 2010.
- [3] GOYAL, M. et al. Artificial intelligence-based image classification methods for diagnosis of skin cancer: Challenges and opportunities. *Computers in Biology and Medicine*, Elsevier, v. 127, p. 104065, 2020.
- [4] CASALE, S. et al. Speech emotion classification using machine learning algorithms. In: IEEE. *2008 IEEE international conference on semantic computing*. [S.l.], 2008. p. 158–165.
- [5] MENEZES, R. S. T. D.; MAGALHAES, R. M.; MAIA, H. Object recognition using convolutional neural networks. *Recent Trends in Artificial Neural Networks-from Training to Prediction*, IntechOpen London, UK, 2019.
- [6] DAELEMANS, W.; HOSTE, V. Evaluation of machine learning methods for natural language processing tasks. In: EUROPEAN LANGUAGE RESOURCES ASSOCIATION (ELRA). *3rd International conference on Language Resources and Evaluation (LREC 2002)*. [S.l.], 2002.
- [7] MOREIRA, A. B. et al. *Multiprova: Software educacional*. 2020. Disponível em: <https://site.multiprova.ufrn.br/>.
- [8] ABADI, M. et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. Software available from tensorflow.org. Disponível em: <https://www.tensorflow.org/>.
- [9] CHOLLET, F. et al. *Keras*. GitHub, 2015. Disponível em: <https://github.com/fchollet/keras>.
- [10] BRADSKI, G. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

APPENDIX A: Confusion matrix and hyperparameter configuration results for letter CNN from section 3.1.

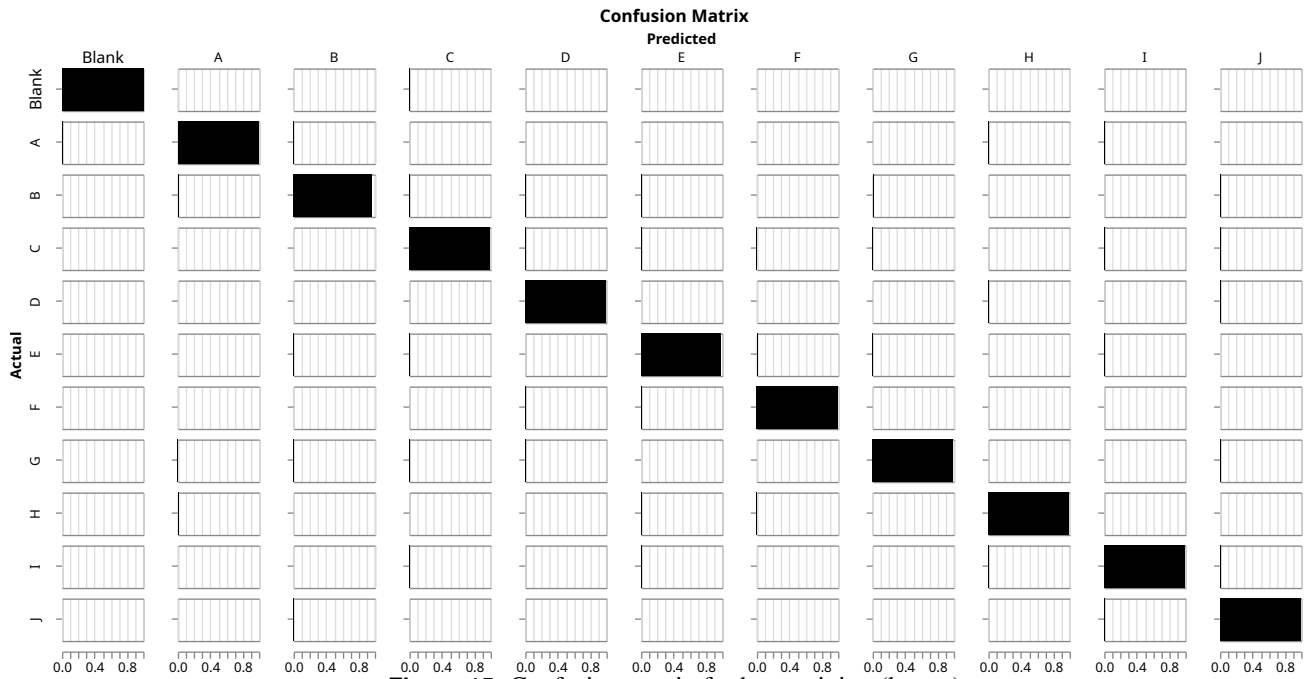


Figure 15. Confusion matrix for best training (letters)

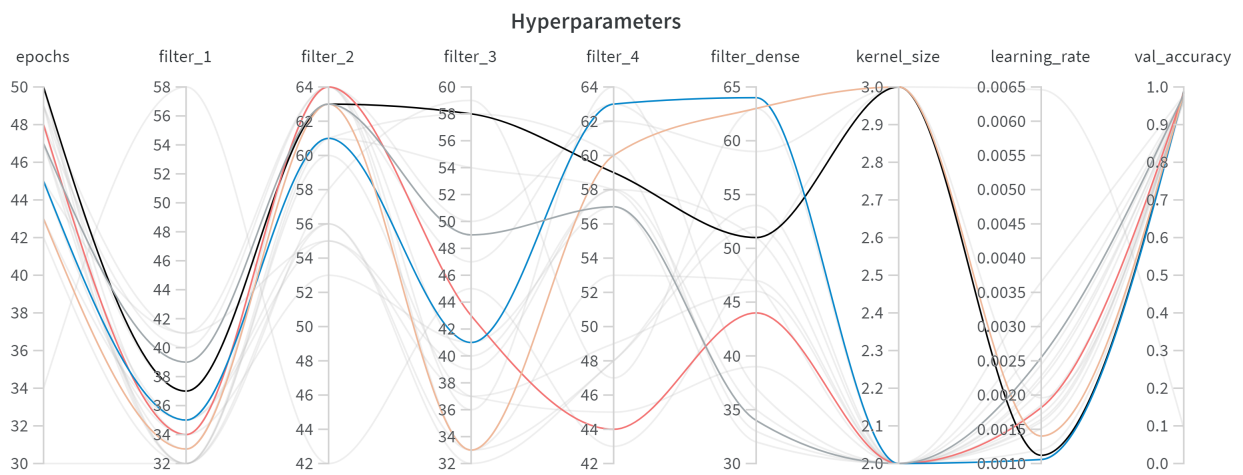


Figure 16. Hyperparameters for 5 trainings (letters)

APPENDIX B: Confusion matrix and hyperparameter configuration results for numbers CNN from section 3.2.

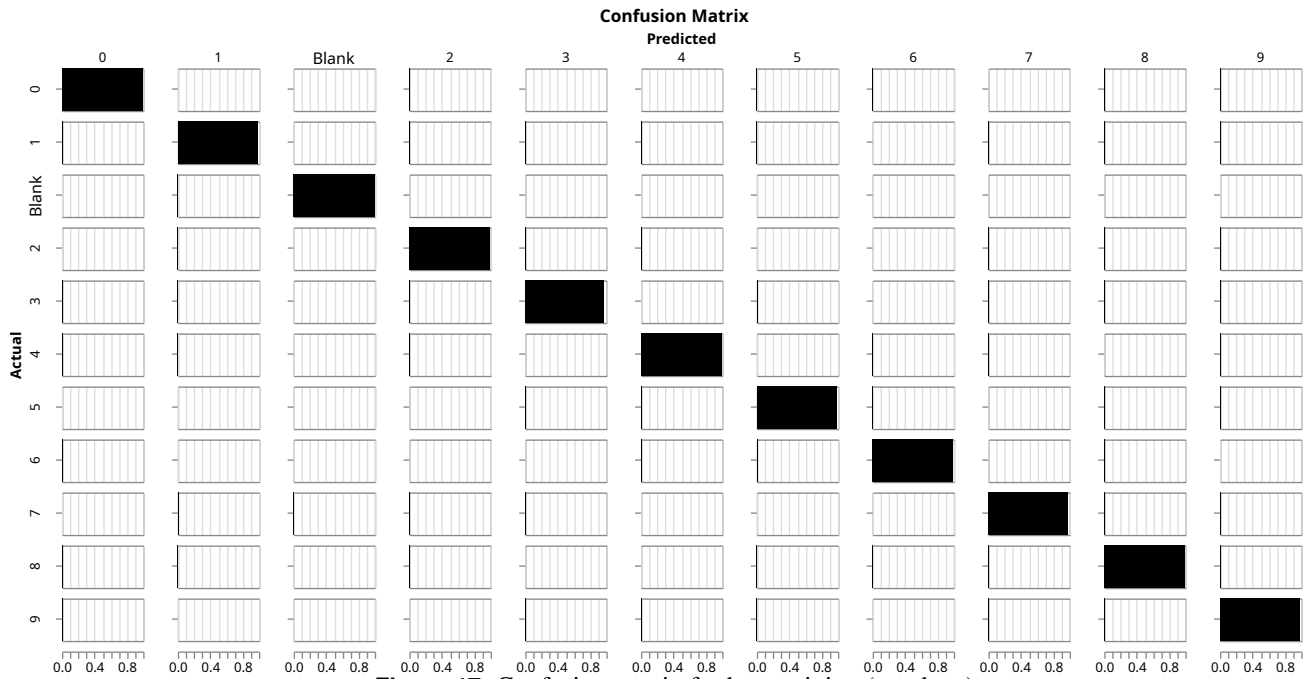


Figure 17. Confusion matrix for best training (numbers)

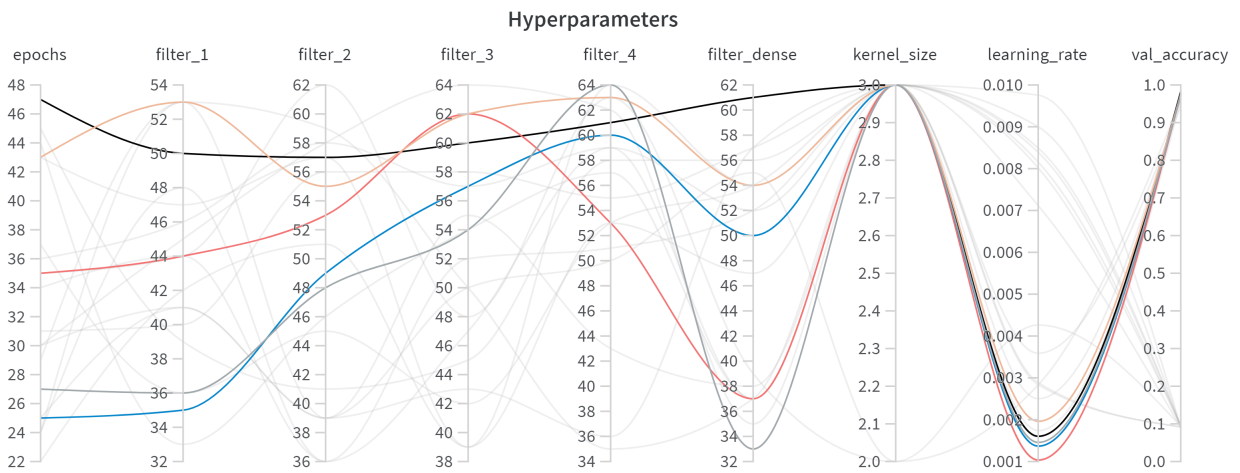


Figure 18. Hyperparameters for 5 trainings (numbers)

APPENDIX C: Confusion matrix and hyperparameter configuration results for true or false CNN from section 3.3.

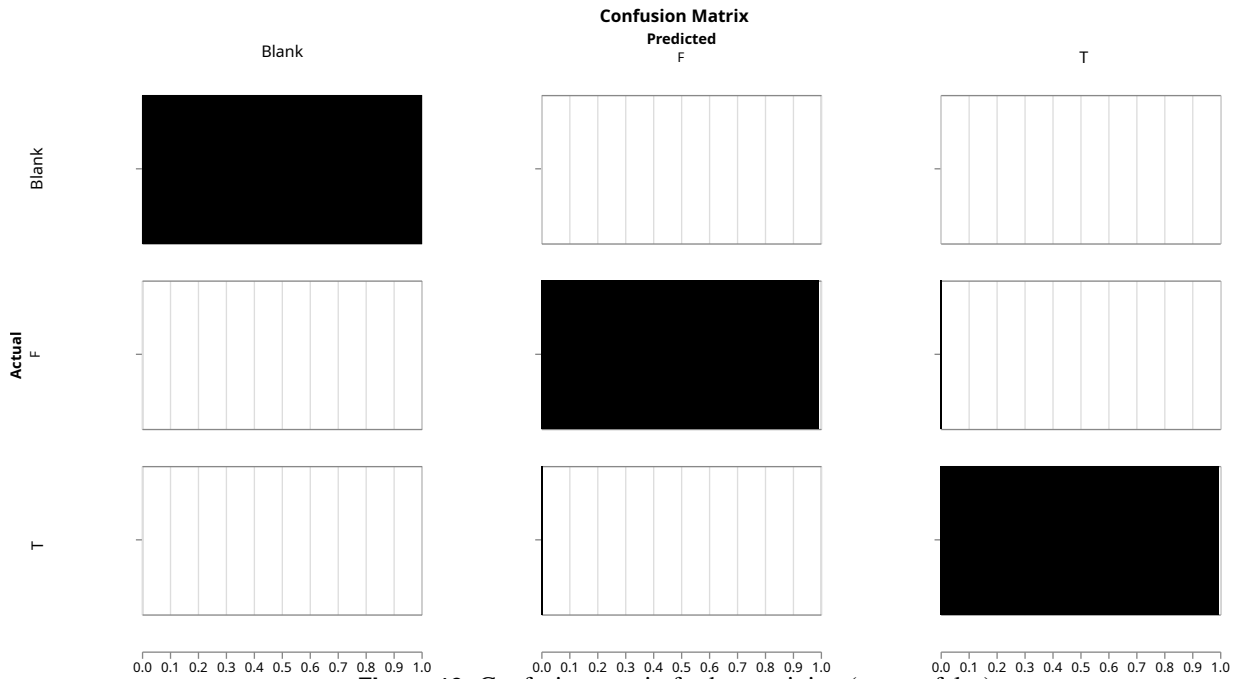


Figure 19. Confusion matrix for best training (true or false)

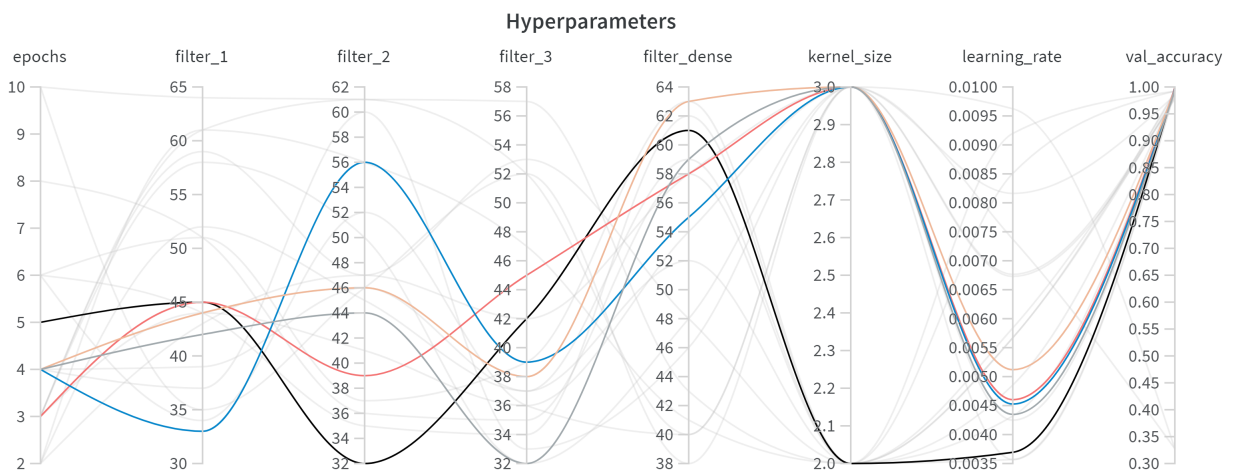


Figure 20. Hyperparameters for 5 trainings (true or false)

APPENDIX D: Hyperparameter importance and correlation results for CNNs from section 3.3.

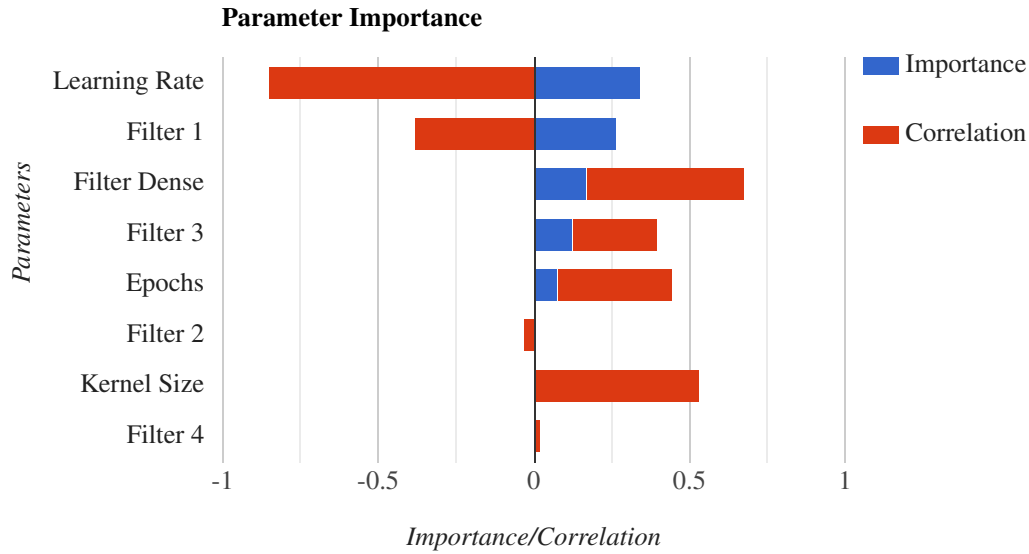


Figure 21. Hyperparameter importance/correlation (letters)

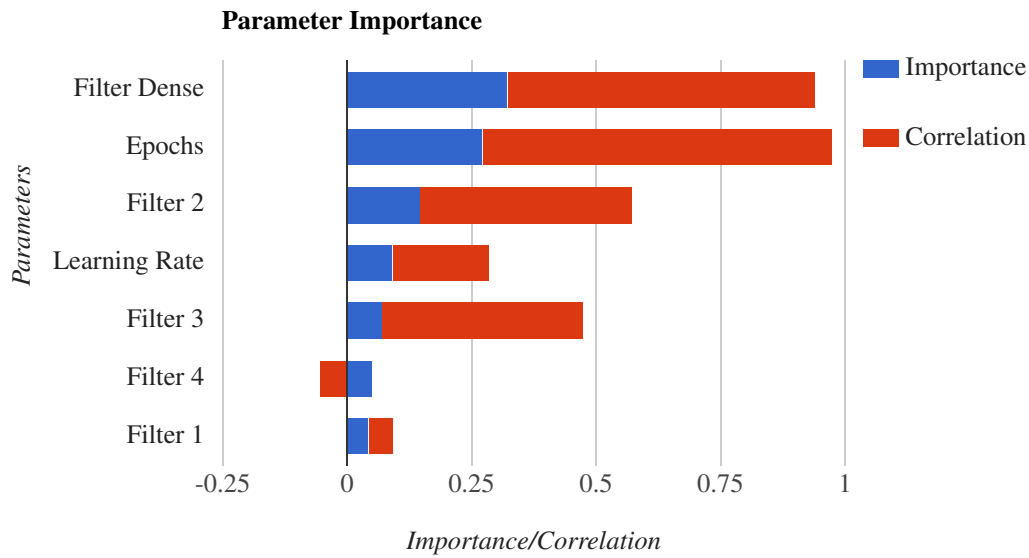


Figure 22. Hyperparameter importance/correlation (numbers)

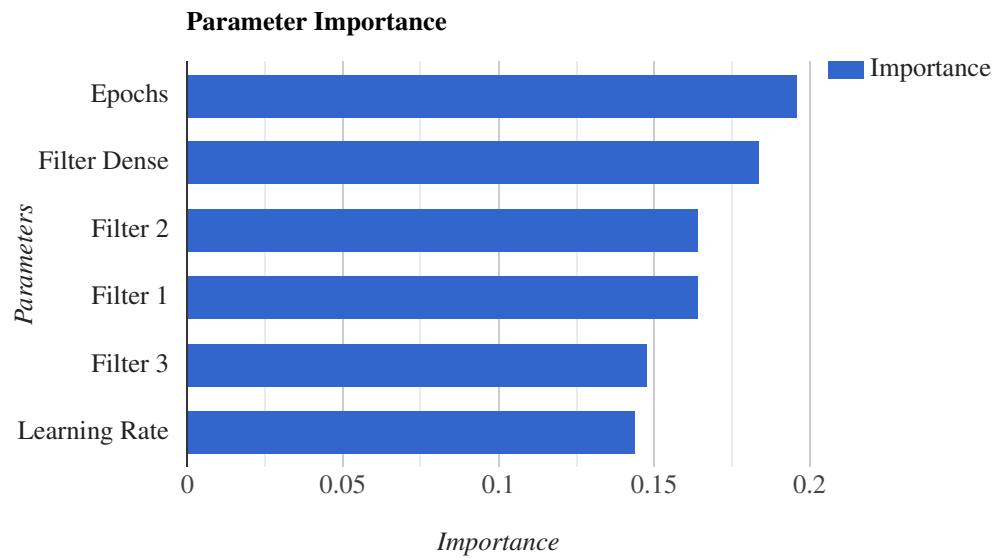


Figure 23. Hyperparameter importance (true or false)