Machine Learning Project
Building a CNN for Rock-Paper-Scissor Classification

Index

Introduction 2

Chapter 1 Data Exploration and Preprocessing 3

1.1. Data description and summary of observations 3

1.2. Image preprocessing and data splitting 3

Chapter 2 Convolutional Neural Networks architecture and training 5

2.1. CNN 1: a baseline model 5

2.2. CNN 2: a more complex framework with hyperparameter tuning 6

2.3. CNN 3: a final architecture 8

Chapter 3 Model evaluation and analysis 9

3.1. Comparing performances using proper metrics: accuracy, precision, F-1 score and recall 9

3.2. Comparing training curves: loss and accuracy curves 11

3.3. Misclassified examples analysis: what are the most common errors? 12

Chapter 4 Models' generalization capability: evaluating performance with a new dataset 13

4.1. Accuracy analysis 14

4.2. Misclassified examples analysis 15

Conclusion 15

Introduction

This project aims to design, develop and evaluate three different architectures of convolutional neural networks of increasing complexity, in order to make predictions about a dataset containing images of hands making the gestures of rock, paper and scissors. Although the task seems rather simple, this type of project allows us to explore the strengths and limitations of convolutional neural networks and address typical machine learning problems, such as finding the right balance between training time and performance or managing the risk of overfitting and underfitting.

The project was split into three main pillars: the first part was devoted to an initial, general overview of the dataset, where the distribution of images, as well as the cleanliness of data was checked. After these first steps, data have been split, and some augmentation techniques have been applied in order to reduce the chances of overfitting.

The second part instead was dedicated to build the CNNs, keeping in mind that their complexity should have increased from model to model. Once built, the networks have been trained on the same training set, and for one of them, hyperparameter was performed, in order to choose the best possible parameters of the predictor.

The third step was then to compare the predictors obtained in the previous part, in order to evaluate their performances and select which was the best among them. In order to do such a thing, several metrics have been computed, and an analysis on the errors most frequently made by all three models was performed. To compare how well the networks were learning and how much were they able to generalize, some plot regarding the loss and the accuracy of the models were produced.

Lastly, all the networks have been evaluated on a new dataset containing different images. This made it possible to verify the generalization abilities of each model, while also providing a powerful tool for checking which of them could potentially suffer from overfitting.

Chapter 1

Data Exploration and Preprocessing

1.1.Data description and summary of observations

To begin exploring the data, a brief resume of the number and category of images was produced. In particular, the entire dataset contains a total amount of 2188 images, divided into three classes: rock, paper and scissor. An important aspect to consider when building machine learning models such as those required by this project is certainly the distribution of the classes of the dataset: in fact imbalanced classes could lead to several problems, including difficulties in the learning phase of the model, which may perform worse in predicting the label of examples belonging to a class with very few data in it. Moreover, some metrics used to evaluate the predictors such as accuracy, could be meaningless if not interpreted correctly. In our case, the classes are distributed as follows:

- Rock: 726 images (33,18%)
- Paper: 712 images (32,54%)
- Scissor: 750 images (34,28%)

Fortunately, as one can easily notice, classes are balanced and distributed fairly evenly, so no special precautions are required.

Next, it was checked whether all images had the same dimensions, another crucial step that allows to train properly the neural networks. The result is that all images have the same dimensions, which are 300 pixels as width and 200 pixels as height, so once again, no particular changes are needed. Last but not least, the presence of corrupted images, or images with incorrect extension, or that cannot be correctly read has been checked, and fortunately, the result is that there are no damaged images in the dataset.

Finally, to verify that everything was working correctly, some random images belonging to the three different classes were displayed. This allows us to proceed with the preprocessing and data splitting phase.

1.2.Image preprocessing and data splitting

In order to comply with one of the requirements that this project aims to meet, the first thing that has been done was to set the parameters that define the new image dimensions. This procedure, called reshaping, is crucial if we want to accelerate the training time of each network: reducing the number of pixels means that the network will have to process less data, and can therefore focus on the most important features of the images, thus reducing the risk of overfitting. Considering that the three hand

gestures (rock, paper, scissors) are quite distinct and that I wanted to avoid excessive distortion of the images, after few attempts, I opted to resize the images to a new size of 150x150.
Another particularly important aspect was to normalize the pixels values, in order to make them with mean zero and standard deviation equal to one. This process is extremely helpful in making the networks more efficient in the training process, reducing the time needed to get the predictor and making the gradients more stable.

Subsequently, data have been split into train and test set, using the ratio of 80/20. Additionally, to ensure greater robustness and predictive capabilities for each model, only the images belonging to the training set have been augmented: this procedure is very helpful to reduce once again overfitting, because we are basically feeding the network with new images that it will have to learn. This technique ensures that models will not focus solely on the main details belonging to the training set, but will receive new data every time, simulating what can happen in real life scenarios. The augmentations applied are the following:

- Rotations (of maximum 20 degrees)
- Horizontal shifts (up to 10% with respect to the dimensions of the image)
- Vertical shifts (up to 10% with respect to the dimensions of the image)
- Zooms (of maximum 20%)
- Horizontal flips

The goal of these simple transformations was to make the predictors be able to generalize better, giving us the opportunity to recognize also different patterns with respect to those included in the dataset.
Another important aspect is that I made sure that the “Shuffle” parameter was set to true only for the training set, so that in each epoch every batch contains different and random images, making each one of them more heterogeneous and reducing the likelihood of overfitting. As mentioned, the test set remains instead unchanged.

Lastly, for technical reasons related to the calculation of the loss, the image labels have been converted to one-hot vectors, so that we can use the categorical cross-entropy when building the convolutional neural networks.

The training set is then composed of 1751 images, while the test set of 437.

Chapter 2

Convolutional Neural Networks architecture and training

The next sections of the project are dedicated to explaining how the models were built, and the decisions made for designing the architectures of each of them. To start, a first simple model was designed, followed by two others of increasing complexity. This approach is particularly helpful for understanding the trade-offs between complexity, training time, and overfitting or underfitting. In fact, we expect simpler models to learn faster, but on the other hand, they may be less accurate and suffer from underfitting. Overly complex models instead, could adapt too much to our data, and suffer from overfitting. At the end of the training, each model and its respective history has been saved, in order to make the result of the evaluation analysis reproducible. We can now see a description of each one of them.

2.1. CNN 1: a baseline

The first network implemented features a rather simple architecture, used as the basis for the two networks developed subsequently. The dimension of the batch the network receives in input is of 32 images, and it is built as follows:

- Convolutional layer 1: this layer applies 32 filters of dimension 3x3; the activation function used is ReLU, the most used activation function for these kinds of tasks, mostly because of its non-linearity.
- Max Pooling layer 1: this layer reduces the dimensions of the feature maps using a 2x2 window and taking the maximum value; this is particularly useful for preserving only the most important information and for reducing overfitting by lightening the network.
- Convolutional layer 2: in this second convolutional layer the number of filters is 64, and the activation function is still ReLU; with this layer we are able to memorize more complex patterns.
- Max Pooling layer 2: this layer reduces again the dimensions of the feature maps obtained from the previous convolutional layer; its utility is the same as the one mentioned above.
- Global Average Pooling layer: this layer takes the feature maps and computes their mean, making then a vector; this was preferred to Flatten in order to reduce the number of total parameters.
- Fully Connected layer: this layer takes the vector and computes combinations of its elements, applying then ReLU as activation function; it has 64 neurons.

- Output layer: this is the final layer, it has three neurons, and its activation function is SoftMax, a function that allows to compute the probability of belonging to each of the three classes.

Adam was chosen as optimizer for updating weights, mainly because of its robustness and versatility, and a fixed learning rate of 0,001 was assigned, which is the standard value used by Adam. With regard to the choice of the loss function, given the multi-class nature of our problem, we chose to use the categorical cross-entropy, one of the most used loss functions for these kind of tasks, which is a function that penalizes more when the model assign a high probability to the wrong class. The metric used for monitoring the performances of the model was the accuracy, the percentage of correct predictions.
The model was therefore trained for a total of 20 epochs without introducing an early stopper, as it was expected that the time needed to train it would not be too long.
From this starting model we expect pretty reasonable performances, due to its simplicity, although it is clear that it can certainly be improved, and this is what we have attempted to do subsequently.

2.2. CNN 2: a more complex framework with hyperparameter tuning

The architecture of the second convolutional neural network was built on the basis of the first, seeking to increase complexity and hopefully improve performance, keeping in mind the risk of overfitting.
The main structure of the network remains in fact the same, but a convolutional layer and another Max Pooling layer were added, in order to let the network learn even more complex details of the images. The network is built as follows:

- Convolutional layer 1: 32 filters of dimension 3x3, with ReLU activation function.
- Max Pooling layer 1: layer reducing dimensions of the feature map, using once again a 2x2 window and taking the maximum value.
- Convolutional layer 2: 64 filters of dimension 3x3, where the activation is ReLU.
- Max Pooling layer 2: same layer as the previous with a 2x2 window, it will help maintaining only the most important information.
- Convolutional layer 3: this final convolutional layer now features 128 filters with same dimensions as the previous ones and uses ReLU as activation function; this layer should allow the network to learn even finer details, improving then its ability to correctly classify the images.
- Max Pooling layer 3: last pooling layer, the window is again a 2x2.

- Global Average Pooling layer: the choice was to use again Global Average Pooling and not Flatten in order to not overload too much the network and reduce the training time.
- Fully Connected layer: in this case the dense layer doesn't have a fixed number of neurons; this because of the hyperparameter tuning explained later, which will help identify the best number of them.
- Output layer: final layer with 3 neurons and SoftMax as activation function.

With this deeper structure we expect to achieve even greater performances with respect to the first network: the two layers just added should improve the final accuracy of the model. On the other hand, this architecture surely increases the computational costs, increasing the training time, and there is the possibility to face overfitting.
Another difference is that for this neural networks hyperparameter tuning with grid search and stratified cross validation has been applied, in order to find the best combinations of batch size, learning rate and number of neurons for the dense layer. This procedure allows us to choose the parameters that guarantee the best performance, as well as reducing the risk of overfitting and underfitting. As mentioned, the idea of the grid search was followed: for each hyperparameter 2 possible values were chosen, and using the structure of the neural network previously described, the model resulting from the combination of each hyperparameter was computed, in order then to compare their performances. The values chosen for the hyperparameters are the following:

- Learning rate: [0.001, 0.0005]
- Batch size: [32, 64]
- Neurons in the dense layer: [64, 128]

The models resulting from the combination of each parameter are then 2x2x2=8. Then, instead of computing each model a single time, a cross validation has been computed, in order to get more robust and reliable results. The training set was in fact split into three folds, so that each model resulting from a combination of parameters was trained for k=3 times, using two of the folds for training and the left out for validation. At the end, a mean of the accuracy of these three rounds was computed, in order to finally evaluate the performances of these 8 neural networks. An important aspect to mention is that, only for the purpose of running the cross validation and getting less variable and more consistent results, data augmentation have been reduced. In particular, zooms and flips have been removed, while the rotation range has been reduced to 5 degrees and the horizontal and vertical flips

have been reduced to 5%. Another crucial point is that, for computational reasons, the epochs were set to 5, in order to reduce the training times of each model.
The result of the hyperparameter tuning suggested that the parameters to use to train the network were the following:

- Learning rate = 0.001
- Batch size = 32
- Neurons in the dense layer = 128

Using these parameters, the convolutional neural network was then trained. As optimizer, both in the hyperparameter tuning and in the final training, Adam was used, and the same reasoning applies for the loss, which is again the categorical cross-entropy, and for the monitored metric which is the accuracy.
The model was trained for 20 epochs but in this case, we added an early stopping, which is a regularization parameter that stops the training if a certain metric does not improve for a chosen number of consecutive epochs. In this case the test loss was set as metric, and 3 as number of consecutive epochs.

2.3. CNN 3: a final and more complex architecture

Once again, the structure of the network developed in the previous step was used to build this last neural network, but some measures have been put in place to try to differentiate the network from the previous one. The main architecture remains then the same, but 3 layers have been added to this version, making it deeper and hopefully improving the classification capabilities. The network features some layers that are identical to the previous model:

- Convolutional layer 1
- Max Pooling layer 1
- Convolutional layer 2
- Max Pooling layer 2
- Convolutional layer 3
- Max Pooling layer 3
- Global Average Pooling layer
- Fully Connected layer
- Output layer

Where all their activation functions, filters, dimensions, and neurons are the same as the previous ones. To this setup the following layers have been added:

- Convolutional layer 4: layer with 256 filters of 3x3 dimensions, capable of learning very complex details; its activation function is ReLU.
- Max Pooling layer 4: another layer to reduce the dimensions of the feature map.
- Dropout layer: a layer that disables randomly some of the neurons of the previous layer, in order to make the model more robust to overfitting; the percentage of neurons disabled in each epoch has been set to 50%.

This particular architecture has been designed keeping in mind that increasing the complexity of the network could significantly increase the risk of overfitting, and for this reason, certain measures were put in place. As first, the main countermeasure adopted was the dropout layer, which is a very helpful regularization technique that helps reducing overfitting, forcing the network to not depend only on some specific neurons. For this reason, it has been placed right after the fully connected layer. Finally, the last things that have been modified are the batch size, which was increased to 64, and the learning rate, which was decreased to 0.0005. These changes have been made in order to make each update more stable, even though the learning process is slightly slower.

Finally, the model was trained, using once again Adam as optimizer, the categorical cross-entropy as loss and the accuracy as metric. The training was planned for 20 epochs, even though an early stopping with three epochs of patience and the validation loss as monitored metric was added.

Chapter 3

Model evaluation and analysis

This final section is devoted to the analysis of the performance of the convolutional neural networks, in order to compare their capability to correctly classify images and their ability to generalize. To do this, the analysis started with the comparison of some specific measurements, followed by the graphical representations of the training and loss curves.

Immediately after, a study of the images misclassified was produced, so that we could understand the most common errors made by each model and verify their strengths and weaknesses.

Lastly, a brief resume of the evaluation is made, making also some considerations about overfitting and underfitting.

3.1. Comparing performances using proper metrics: accuracy, precision, F-1 score and recall

As a first step, four metrics were computed: accuracy, precision, F-1 score and recall. Each one of them resumes specific information about the quality of the model. Accuracy is the percentage of

correctly classified images with respect to the total number of images in the test set and gives a general overview of how good is the model in predicting classes. Precision is, for a given class, the number of images correctly predicted with the label of that class, over the total number of images classified with that class. This metric is particularly useful if we want to keep under control when the model predicts false positives. Recall measures again, given a certain class, the number of images for which the correct label is assigned, but this time over the total number of images belonging to that class. This one is instead particularly useful if the focus is on the false negatives. In the end, the F-1 score is instead a metric obtained by precision and recall, so it is useful if you want to resume both the information of the measures previously described.
The measurements obtained are resumed in the table below:

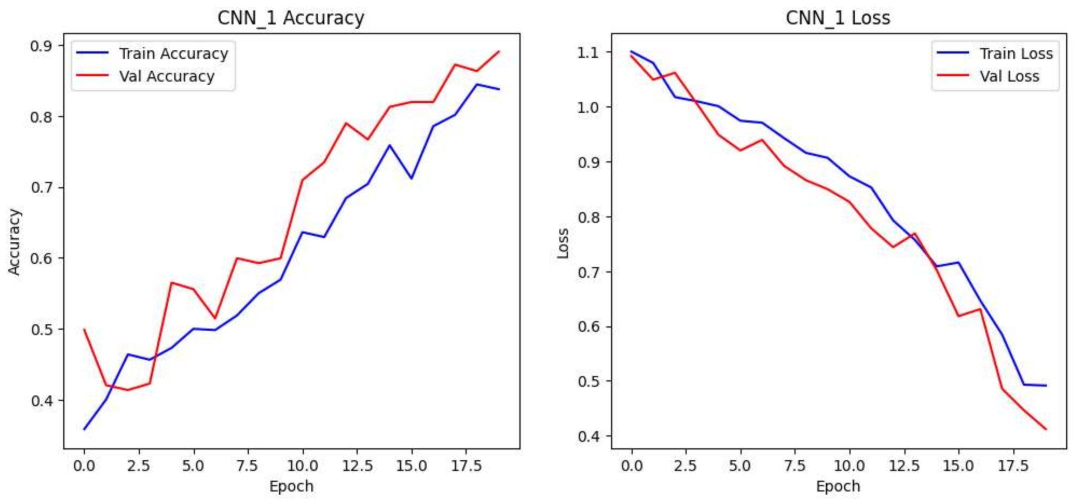
Model	Accuracy	Precision	Recall	F1-score
CNN_1	0.89016	0.89209	0.89064	0.89023
CNN_2	0.97254	0.97309	0.97251	0.97252
CNN_3	0.98169	0.98200	0.98160	0.98163

As one can immediately notice, for each model the metrics are very close one to the other, and this is mainly due to the fact that classes are very balanced.
The first neural network achieves pretty good results, obtaining a mean of 89% for every metric. This clearly shows that, despite its small size, it remains a fairly good model to use as a baseline but still leaves room for possible improvements. In fact, with the second neural network, the improvements achieved are remarkable: deepening the network with a convolutional layer and a pooling layer significantly increases performance, moving to an average of the metrics of 97%. This confirms our expectations, namely that by increasing complexity we can enable the model to learn even more sophisticated features, improving the overall performance. Lastly, the third neural network still gets some improvements, even though in this case they are marginal. This suggests that we are probably facing a point of performance saturation, and increasing even more complexity would not be very useful. So, this last model could adapt too much to the data and suffer from overfitting, even if by this metrics we cannot be sure. For this reason, this aspect will be explored in depth in the analysis of the training curves.
Overall, we can observe a positive trend in terms of performance: as complexity grows, also the metrics tend to increase, even if at a decreasing rate. This shows that increasing the depth of the network contributes up to a certain point, beyond which performance becomes more stable, and the implementation of other techniques to avoid overfitting and get even better results can be considered.

3.2. Comparing training curves: accuracy and loss curves

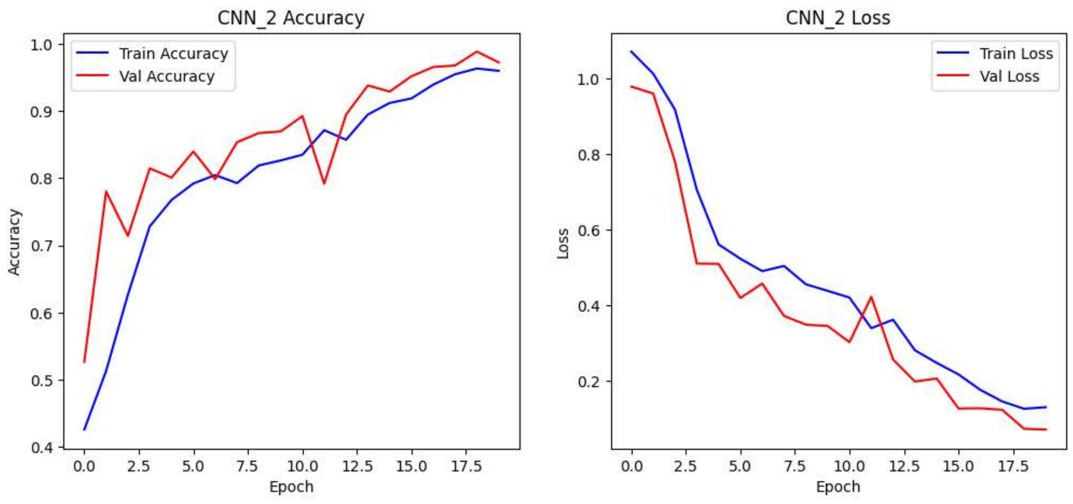
The next step is to analyze the curves of accuracy and loss, to see their evolution epoch by epoch and understand if the models suffer from underfitting, overfitting, or none of the two.

The curves of the first convolutional neural network are shown below:



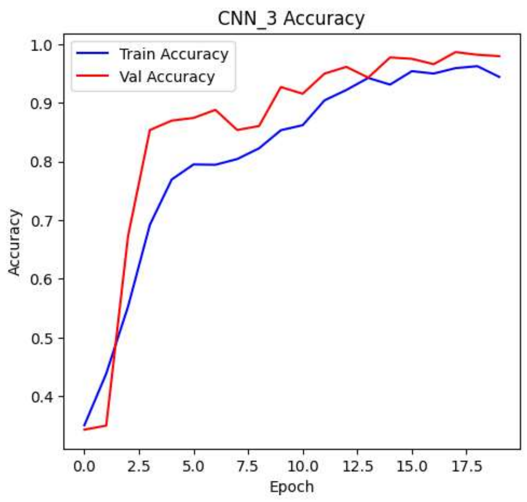
For the first model, we can see that the curves of accuracy of both training and validation, tend to grow gradually and roughly at the same rate. The discrepancy between the two curves is limited, suggesting that the model is not suffering from overfitting. The same reasoning applies to the loss curves, which lead us to the same conclusions. As mentioned before, the two accuracy curves reach a score of 89%, meaning that the model is balanced but too simple.

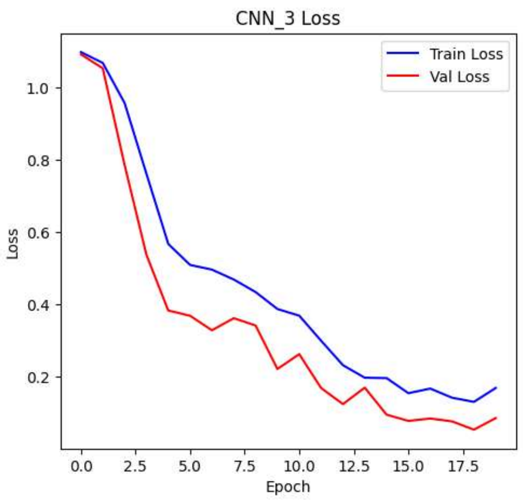
For the second model, the following results have been achieved:



In this case the increase in complexity and therefore in performance is evident. The accuracy curves seem to flatten out towards the last epochs, after following a very similar growth trend in the first ones. Moreover, growth is much faster than in the first network, meaning that with this setup the model is able to learn rapidly the key features of the images. Consistently, also the decrease of the loss curves is very fast, and at the end they seem to converge. Also for this model, the risks of overfitting and underfitting seem to have been avoided. The architecture then looks well developed, ensuring a high level of performance without overfitting.

The third and last model allowed us to get these plots:





For this last network, the pattern of the curves appears to be quite similar to the one of the second network, making clear that the increased complexity of this architecture does not translate into a significant increase in performance. Accuracy grows rapidly and also the loss decreases very fast in the first epochs. The curves are once again close one to the other, suggesting that the model is stable, even if this fast convergence to elevated levels of accuracy, could maybe lead to overfitting if the model is trained for more epochs.

An important thing to notice is that, for every network, the results of the validation (both accuracy and loss) were better than the training. This is an unusual behaviour, which could be mainly caused by the data augmentation applied in the training set, which enables the network to learn using more complex images than those used in the validation process.

3.3. Misclassified examples analysis: what are the most common errors?

The following step is related to the analysis of the mistakes made by each network, in order to discover strengths and weaknesses of each one of them.

The misclassified images are listed below:

```
CNN_1 - Missclassified images: 48

CNN_1 - Errors table:

Predicted  paper  rock  scissors
True
paper          0     7         9
rock           9     0         0
scissors      18     5         0
```

```
CNN_2 - Missclassified images: 12

CNN_2 - Errors table:

Predicted  paper  rock  scissors
True
paper          0     4         2
rock           1     0         0
scissors       1     4         0
```

```
CNN_3 - Missclassified images: 8

CNN_3 - Errors table:

Predicted  paper  rock  scissors
True
paper          0     3         2
scissors       1     2         0
```

The first neural network shows that the most misclassified class is the scissor class, which is frequently predicted with paper. Other frequent errors are paper predicted as scissor, and rock predicted as paper. These mistakes suggest that this model, due to its simple structure and consequently of its lack of filters, could struggle in recognizing when a hand is fully open, or in any case makes some mistakes in recognizing the contours of the hands. In fact, it seems that the model is better in classifying a class like rock, which is the one with the simplest pattern that doesn't require to recognize finger shapes.

This problem is partially avoided with the second CNN: the more complex architecture allows the network to learn more difficult patterns and to make less mistakes. The most common errors are paper and scissors classified as rock, which suggests that sometimes the model struggles in recognizing some details of the fingers. Overall, there is a significant improvement, with the number of misclassified images falling from 48 to just 12.

Finally, the third neural network further reduces the number of errors, but not drastically: of the eight mistakes made, five of them are the same as the previous ones. This confirms once again that the more complex structure of this network does not help too much in reducing the number of errors the model is used to. One last important thing to mention is that we get zero misclassified rock images, making clear that the only weakness of this architecture is probably the classification of particularly complex images of open hands. However, putting aside this minor flaw, the model achieves extremely high performances, being probably the better in generalizing.

Chapter 4

Models' generalization capability: evaluating performance with a new dataset

This last section aims to understand which model achieves the greatest generalization capability, using a dataset of images personally taken. The pictures used for this kind of task were taken using a background with a similar tone to that of the images used in the training process, while the hands were placed in different positions in order to challenge the networks. Moreover, some details such as rings, bracelets and watches have been added, in order to introduce some variability and noise. This was crucial to verify if the networks were able to classify hand gestures focusing on the essential features, and to assess their robustness.

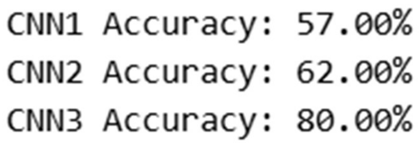
The dataset is composed of 100 images, and the distribution of the classes was taken similar to the one of the training set. The number of pictures for each class is the following:

- Paper: 33
- Rock: 35
- Scissors: 32

Once collected, the images have been preprocessed following the same steps of the test set: as first they were rescaled, and after that they were resized to 150x150, which are the same dimensions used for training the networks.
Then accuracy for each model has been computed, together with the number of misclassified images.

4.1. Accuracy analysis

The accuracy metrics obtained for each model are resumed below:

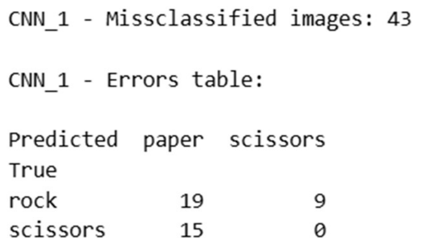


```
CNN1 Accuracy: 57.00%
CNN2 Accuracy: 62.00%
CNN3 Accuracy: 80.00%
```

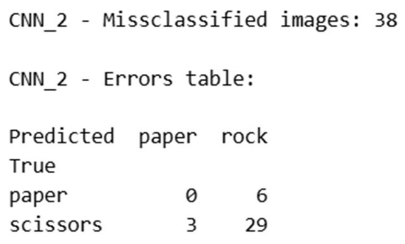
As one can immediately notice, performance has dropped significantly, and this is quite common when evaluating a model on a new dataset that may contain images with slightly different features than the ones the network has been accustomed to. In our case we can see that, as expected, the first model is the one with the lowest performance: this suggests that its simple architecture doesn't allow to recognize the most important features of the hand gestures, making it struggle to recognize peculiar patterns in a new and more complex dataset.
With the deeper architecture of the second neural network the ability to generalize improves although not by much, contrary to the notable increase in performance that was observed during the training phase. This result shows that this model also has important weaknesses in generalizing and classifying images never seen before.
Lastly, the third architecture is the one achieving the highest accuracy, equal to 80%. This shows that, contrary to what is suggested by the slight increase in performance obtained during training, the model with the deepest architecture is also the best at generalizing and also obtains rather good performances being the more robust among the three.
The results obtained make immediately clear that, in this specific case, the increase in complexity and depth of the architectures helps the networks learn increasingly complex details, thus making them less dependent on training data and better at generalizing.
However, it is also important to notice the significant drop in accuracy of each model, which may suggest that all networks have learned very well the main characteristics of the dataset with which they were trained. This can be a clear sign of overfitting, which in the future could be fixed with specific techniques, such as for example introducing even more regularization.

4.2. Misclassified examples analysis

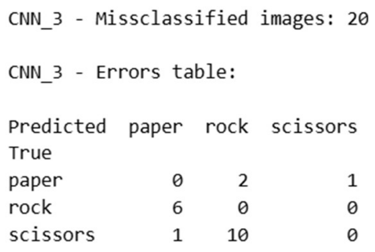
The table of misclassified examples obtained evaluating the model on the new dataset is the following:



```
CNN_1 - Missclassified images: 43

CNN_1 - Errors table:

Predicted  paper  scissors
True
rock          19         9
scissors      15         0
```



```
CNN_2 - Missclassified images: 38

CNN_2 - Errors table:

Predicted  paper  rock
True
paper          0     6
scissors       3    29
```



```
CNN_3 - Missclassified images: 20

CNN_3 - Errors table:

Predicted  paper  rock  scissors
True
paper          0     2         1
rock           6     0         0
scissors       1    10         0
```

The first model, similarly to what we have seen in the previous misclassification analysis, still makes a lot of mistakes in correctly classifying the scissors class, often confusing it with paper, exactly as happened with the original dataset. However, with the new images, the network struggles more in recognizing the rock class, which is predicted as paper 19 times. On the other hand, now the network does not make any mistakes on paper images. This seems to show a new weakness of the network, namely distinguishing the compact form of the fist from the form of the open hand, making clear its difficulties in grasping details.

Moving on to the second neural network, the most common errors are the same as before: paper and scissors are often predicted as rock, but this time, the number of misclassified scissors is way higher. Similarly as before, the model very often confuses the initial compact part of the hand in the gestures of the scissors, with the compact part of the rock hand gesture. This compact part seems then to be very crucial in the classification process, making then clear that the network should be trained better to recognize also the fingers, which are a crucial feature for the scissors images.

The last network maintains some of the most common errors, such as scissors often predicted with rock, but also confuses some rock with paper. Despite being the best in terms of performance, the network still makes some of the mistakes that characterize the two previous models. In particular, the difficulty in distinguishing the compact part of the scissors from that of rock persists, and this is its main point of weakness. Furthermore, part of the residual errors are committed due to the difficulty in recognizing the compact forms of rock and paper. However, despite these errors, the network certainly remains the most robust and the best to generalize, although some improvements to make up for these weaknesses can still be put into practice.

Conclusion

To conclude our analysis, we can now summarize what we find out throughout the project.

The first convolutional neural network provided a fair baseline, with a rather simple structure. It achieved good results when evaluated on the original dataset, showing a solid ability to learn features

of the images. Unfortunately, when tested on different pictures, it showed some point of vulnerability, above all the difficulty in recognizing compact shapes, making it the worst model at generalizing.
The second model instead, with its more complex architecture composed by one more convolutional and pooling layer, has made a clear leap forward in terms of performance, making it better in capturing details of hand gestures. However, when tested with the new dataset, it provided disappointing result, struggling in recognizing patterns of fingers, and showing a sufficient although not exciting ability to generalize.
Finally, the third model, the one with the most complex and deeper structure, achieved the best results. Despite the improvement in performance on the original dataset was not very high, this network showed a good capability of generalizing on the new data, even if it still made some errors similar to the ones that characterize the previous networks. Overall, this was the best model among the three.
To conclude, all the models showed particularly good performance on the original dataset, improving as the complexity increased, while they showed a decline when tested on new images. This makes clear that the networks, even if all of them learned the data they were trained on quite well, could suffer from overfitting. This problem could be faced in the future by implementing further data augmentation or regularization techniques, or even by training the models with more different images. In the end, there is still room left to improve the models, and make them even better at generalizing and more robust, to better adapt to new and never seen data.