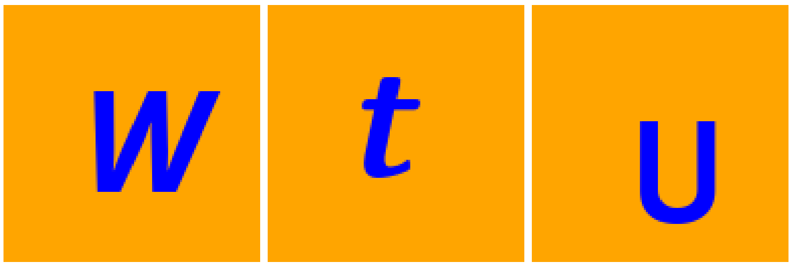
FINAL WORK AUTOMATIC LEARNING 2

1. Descriptive statistics

The set of images has a total of 10,000 images measuring 128x128 pixels. In addition, the images have 3 channels, as they are in colour.



There are 52 different categories. In the following table we can see the number of images for each category:

A	B	C	D	E	F	G	H	I	J	K	L	M
199	178	210	179	180	197	186	183	201	194	203	185	199
N	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**
219	183	207	172	196	185	184	190	219	170	199	207	198
a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**	**j**	**k**	**l**	**m**
211	160	199	193	180	184	181	198	172	222	211	175	186
n	**o**	**p**	**q**	**r**	**s**	**t**	**u**	**v**	**w**	**x**	**y**	**z**
194	212	189	202	171	213	182	195	217	203	177	184	166

The letter with the most images is the letter **j** with 222 images. We also find categories such as the letter **b**, with 160 images, which is the letter with the fewest observations. Although they are somewhat unbalanced, we do not think that this will affect the construction of the model.

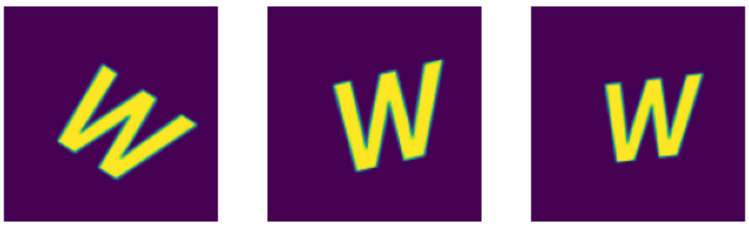
2. Preprocessing of the data

We have carried out a pre-processing of the images before starting the training. This pre-processing consists of:

1. We have only taken one of the channels of the images, as the colour of the images is irrelevant, thus reducing the size and the future number of parameters.
2. Standardisation of data.
3. Data augmentation, to increase the number of images by making modifications to the images we already have. These modifications are zooms and small random rotations. We can see an example below:

Original image

Example Data augmentation



We had to convert the data labels to numerical format in order to be able to start the training.

3. Experiments prior to the final model

We have carried out a technique called Neural Architecture Search (NAS) with the aim of finding the architecture that best fits our set of data without having to do all the architectures manually, i.e. automating the creation of the neural architecture.

This technique is used to adjust models with different convolutional layers, max pooling and FC layers at random, and validates them in order to be able to compare and choose the best model that has been found.

One of our concerns is to create the search space. We know that, theoretically, there are infinite possibilities of how many configurations exist. The number of neurons in each hidden layer can be any positive integer number. So, in our search space for the convolutional layers we have opted for the following possibilities:

- In the first part, we use for the first time the NAS technique, and it takes a number of convolutional layers between 1 and 3, and for each of the layers it takes different values of feature map between 32, 64 and 128. It can take different values for each of the layers (if it chooses more than one).

- In the second part we do the same but now the number of convolutional layers can be a number between 0 and 3. If you choose a value higher than 1, you must also choose the number of feature maps between 32,64 and 128.

- In the last part, choose the number of FCC layers that will be in the model, between 1 and 4. In this layer, we modify the discrete values of nodes that the model can select, now the model can select a number of nodes for each of the layers chosen between the following discrete values: 64, 128, 256, 512 and 1024.

After running the algorithm and adjusting different architectures, we obtained the model that you can see in the following section.

4. Description of the definitive model and its parameters

Our definitive model is a CNN model, which today is a kind of artificial neural network that is more efficient and better at classifying images.

The CNN models have a convolutional part, whose final objective is to extreme the characteristics of each image by compressing them in order to reduce their initial size. These models have two parts, a convolutional part and a classification part. In the convolutional part, the provided image passes through a succession of filters, creating new images, these new images are concatenated into a vector of characteristics. In the classification part, the feature vector is the input composed of fully connected layers called multilayer perceptron (MLP). The objective of this part is to combine the characteristics of the vector to classify the image.

After doing a brute force search with the previous method, the definitive model is as follows:

Layer		Feature Map	Size	Kernel Size	Stride	Activation	Paràmetres
Input	Image	1	128x128	-	-	-	-
1	Average Pooling	1	42x42	4x4	3	-	-
2	Convolution	128	40x40	3x3	1	ReLU	1280
3	Convolution	128	38x38	3x3	1	ReLU	147584
4	Convolution	128	36x36	3x3	1	ReLU	147584
5	Max Pooling	128	18x18	2x2	1	-	-
6	Dropout	-	-	-	-	-	-
7	Convolution	128	16x16	3x3	1	ReLU	147584
8	Convolution	128	14x14	3x3	1	ReLU	147584
9	Convolution	128	12x12	3x3	1	ReLU	147584
10	Max Pooling	128	6x6	2x2	1	-	-
11	Dropout	-	-	-	-	-	-
12	Flatten	-	4608	-	-	-	-

13	FC	-	128	-	-	-	589952
Output	FC	-	52	-	-	Softmax	6708

As can be seen in the model, we have applied a dropout to the convolutional layers, which randomly omits with a probability of 50% neurons during the training process. In addition, we have applied the Average and Max Pooling techniques to reduce the size of the images and, thus, reduce the number of parameters in future layers.

By using data augmentation and dropout techniques, we have managed to avoid model overfitting.

The loss function when compiling the model was **Cross-Entropy**. For the optimisation algorithm we used the **Adam** optimiser with a learning rate of **0.01**.

Finally, the number of **epochs** to train the model was 200.

5. Description of the parameters

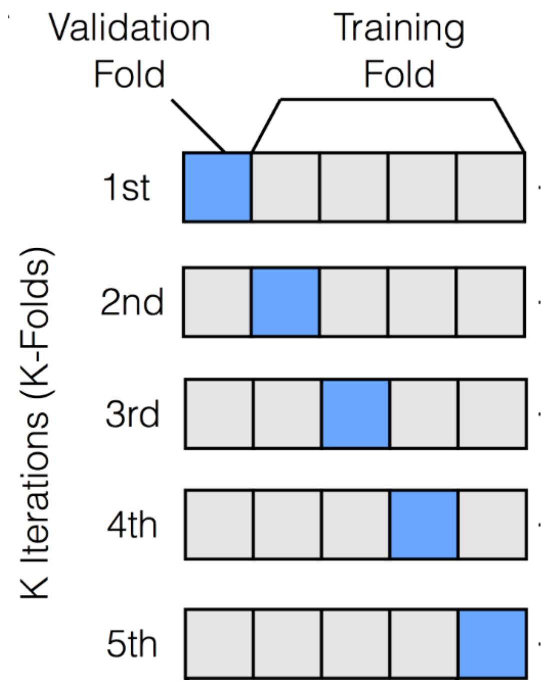
The table in the previous section shows the parameters in each layer. In total, the model contains **1,335,860** parameters. By implementing the max-pooling and average-polling layer in the model, we reduce the computational cost by significantly reducing the number of parameters to be captured.

6. Estimate

Based on our model, the accuracy when running the model on the set of test images we will be given will be **98.1%.**

7. Justification final estimate

The final estimate of the model has been calculated using **K-Fold Cross Validation**, which consists of dividing the data set into k groups, and calculating the model k times where each time one of the groups is used as a test and the model is trained with the remainder, as can be seen in the following image:



In our case, we have used k = 4. Therefore, my estimate is given by the mean of the precision of each of the models that have been validated with the set of test data.

8. Best and worst ranked letters

In the following table we can observe the precision in percentage of the model for each of the letters, in order to observe the categories where the model is most in error.

A	B	C	D	E	F	G	H	I	J	K	L	M
98.5	100	99.1	97.8	100	99	98.9	98.9	85.1	100	99.5	100	100
N	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**
99.5	95.6	99.5	99.4	99.5	99.4	100	98.4	97.3	97.6	99.5	98.5	99.5
a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**	**j**	**k**	**l**	**m**
98.1	99.4	99	98.4	97.8	99.4	98.9	98	98.2	99.1	98.1	73.7	98.9
n	**o**	**p**	**q**	**r**	**s**	**t**	**u**	**v**	**w**	**x**	**y**	**z**
98	99.5	97.3	99	97.6	99.5	98.3	99	98.6	98	98.3	97.3	97

As we can see, most of the classes have a very high accuracy and, in fact, the accuracy of the model for some classes is even higher than 1. However, there are others that are less accurate, such as the lowercase letter L (**l**) and the uppercase letter I (**I**), which have accuracies of 73.7% and 85.1% respectively, as they are very similar and, without any apparent context, the model confuses them with each other.

On the other hand, as we have already said, there are several letters whose accuracy is 100%, as is the case of the letters **B**, **E**, **J**, **L**, **M and T**. The remaining letters have very high accuracies.

9. Observations

Some of the possible improvements of the model would be to obtain a larger set of data and to balance all the categories. One of the letters in which the model fails the most is the lowercase letter L (**l**), which corresponds to one of the letters with the fewest images in the dataset. Perhaps by increasing the number of images of this letter we can improve its accuracy.