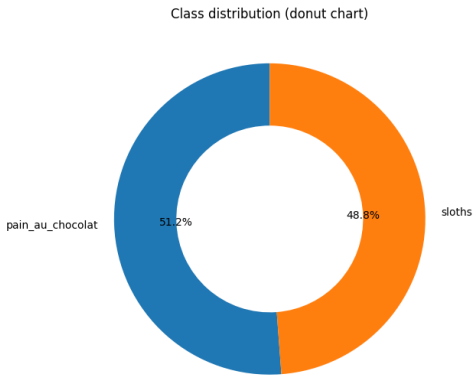
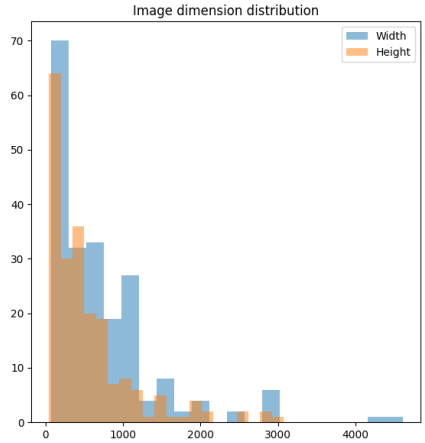
GRAPHICAL RESULTS REPOST

The chosen problem and the Data

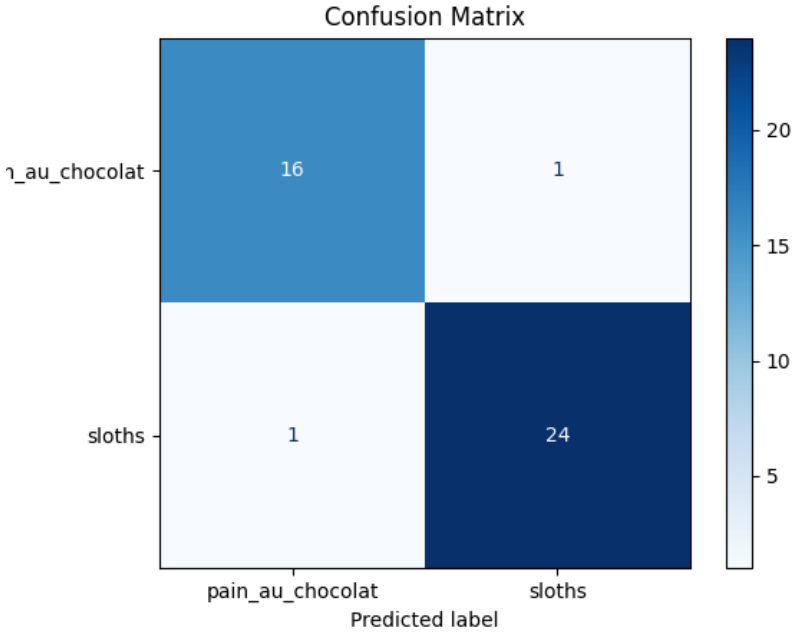
The problem we have chosen to work with, is a binary image classification with a dataset consisting of 209 color images with different sizes split almost evenly among 2 classes: sloth and pain au chocolate. This funny problem, childish at first glance, is actuary pretty tricky as both: the sloths faces and the chocolate buns front side are quite similar (as can be seen in the image subsample). Our first step when we first saw this problem (after laughing a bit), was to check the dataset data. We consider really important to check what shape the images had and how did they look like. Also we wanted to analyze if the color of the images really mattered or if we could just transform them into gray space. In order to represent the data, we have used a histogram to graph the image size distribution, a ring graph for the class proportions (because we have been told not to use many bar plot or any pie chart) and we have used imshow so some pictures from both classes can be seen.

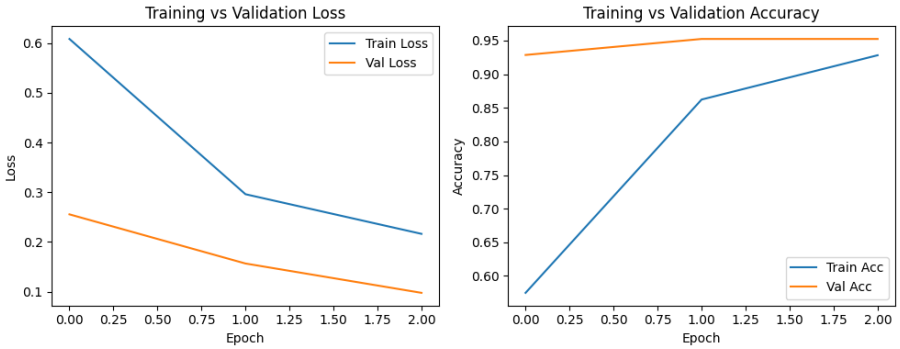


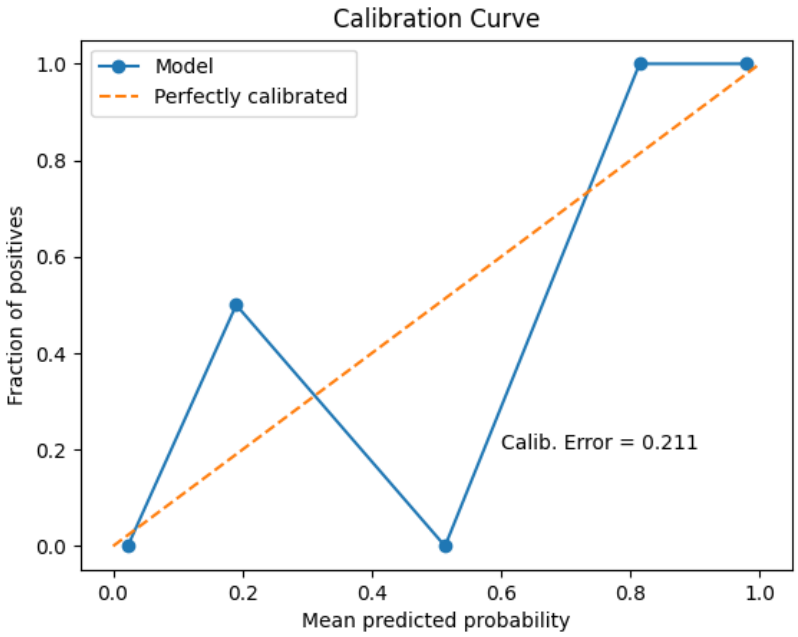


The class distribution shows us that the classes are indeed evenly balanced. The image dimension histogram highlights that the images have multiple different sizes, so some transformations must be made in order to feed them to the model. This, however, might arise new challenges (like deforming or reduce the image resolution) which can make it harder for the model to predict correctly some of the images. Finally, the batch of pictures shown, makes us realize which kind of patterns could make the model fail a classification with this problem (as said earlier, the front side of the pain au chocolate and the sloths faces are quite similar). ### The model The model resizes all the images to the same dimension (128x128 pixels), normalizes them and splits them into train and validation using an 80-20 split. The model is a CNN consists of three convolutional layers, each followed by a ReLU activation function. Then the result is flattered and passed through a simple linear hidden layer and then outputs a prediction. Data loaders are used so the training cycle can mix the Adam optimizer through 3 max epochs with batches (size 32 for train and 4 for val, more or less matching the original dataset ratio). A custom callback is used to store the loss and accuracy values for future use in the graphics section.

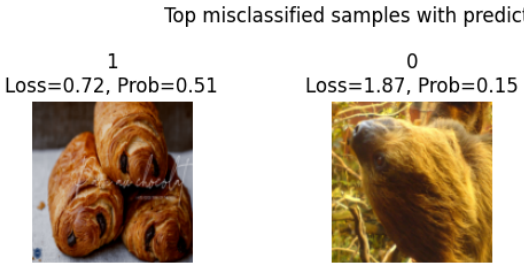
Results The graphs chosen to plot the results are a confusion matrix for the Precision and Recall, the line graphs for loss and accuracy comparison between train and val, and the calibration curve to check if our model is trustworthy.







The versus plots show how the train evolution though the epochs, converges towards an accuracy higher than 0.9. In order to get the same number of values for both sets, we have deactivated the sanity check in validation (after confirming it had a regular value). The validation results start close to where the train ended improving and reach a 0.95 accuracy. This proves that our model isn't overfitted, but instead generalizes really well. The most likely reason for getting a higher validation accuracy than in training is that our dataset is quite small and therefore it's easier to get a higher percentage of correct predictions. The results of this predictions can be seen in the confusion matrix of the validation set. Only 2 out of the 32 instances are wrongly classified, each from a different class (which also serve us as proof or the well balancing of our model). This leaves us with 94% of precision in the chocolate bun class and 96% in the sloth class. For a better understanding of our model, we used imshow to check on the images of those two mislabeled samples to try to guess why they were misplaced.



A sloth was predicted to be a pain_au_chocolat with 0.84 confidence and the opposite happens with a 0.52 confidence. This 2 errors have a great repercussion into the calibration curve. The perfect result for this kind of graph would be a smooth line from 0 to 1, but if the points 0.16 and 0.74 are checked on the x-axis (mean predicted probability), we see a higher value than expected in the first one and a lower value in the second one. This is the consequence of those wrong classifications (again, this isn't a big problem, but the small dataset size increases the error relevance). ### Conclusion As a final conclusion, the accuracy scores and the precision tell us that our model classifies quite well the different images (making it a good option for similar problems). Thanks to this graphs we can also check the data of different parts of the process in an intuitive and visual way, finding possible errors and hardships for the model that otherwise would be difficult for us to notice. This allows us to better comprehend the work the model has performed and to check that cohesion exists among the data shown on the different plots. As a future upgrade, we could try to use a bigger dataset and try other preprocessing techniques to correctly classify those 2 mislabeled samples.