Team: Unilayer Perceptron

Image Classification Project

Deep Learning

Carlos de la Torre Ortiz

May 2020

1 Task Description

The task is to learn to assign the correct labels to a set of images. One image may belong to many classes in this problem, i.e., it's a multi-label classification problem. Some of the images do not belong to any of our classes.

The images are originally from a photo-sharing site and released under Creative Commons-licenses allowing sharing. The training set contains 20,000 images resized to 128 × 128 px.

2 Data Loading

Before data exploration, data needed to be loaded into appropriate data structures that would allow efficient manual inspection. A shell script was used to join the image names with their label. As images may be assigned more than one label, this resulted in image name duplications. This early pre-processing file was then loaded in Python to compose the final pandas data frame in the long (tidy) format. The data frame would have all image identifiers as the index, and each unique label as a column. Labels were hot-encoded as a value of "1" if they were present in the image or "0" if not.

Looking back, it may have been easier to split the images pull into directories named after labels and use Pytorch built-in torchvision folder data loading to make the custom dataset. However, it was unsure for now how it would handle images having multiple labels, and perhaps easier to experiment with having one defined data frame which can be easily sliced.

3 Data Exploration

Preliminary exploration was performed to spot potential problems with the dataset. Many images lacked a label, with only about half the images having at least one label assigned. Furthermore, many images were clearly mislabeled. There are workarounds for this issue to manually correct the labels, even as human-in-the-loop models for identifying likely mismatches. However, given the limited manpower of the group, and that all groups would face the same handicap, it was left as a dataset limitation.

The further exploration involved label counts. Somewhere very overrepresented, others very underrepresented (imbalanced dataset), and some class membership was not clear. Label "people" was clearly overrepresented, followed by "male" and "female". "Baby" was the most underrepresented label. As a result, misclassification biased towards the overrepresented class (mostly "people") and against the underrepresented class (mostly "baby"). In an extreme case, it could lead to a situation in which everything is classified as people. Misclassification could occur as the product of the wrongly assigned labels. Depending on how to solve the problem (e.g. balancing), could lead to data scarcity. As many (about half) of the images lack a label and these have to be managed, label probabilities have to be adjusted accordingly.

Regarding label ambiguity, label “people” could mean only adults, or also includes babies; if it is used only for groups of people or also a single person. “Male” and “female” could also be theoretically assigned to dogs, birds, or even plants. “Portrait” is not clear as it might be exclusively a painting, a way to take a photograph, or the more general definition of graphics in which the main feature is a face. Rough disambiguation was done by manually sampling the dataset and checking the labels. The conclusions for this dataset are the following:

- People label is age-independent.
- People label is number-independent.
- Only humans are classified as male or female.
- Portrait label follows the general definition.

4 Dataset and Data Augmentation

I followed the officially listed PyTorch tutorial on writing custom datasets, data loaders, and transforms `https://pytorch.org/tutorials/beginner/data_loading_tutorial.html`. Several data augmentation methods were also developed foreseeing the challenges of the imbalanced dataset. In particular, a resize method was developed for early prototyping. The dataset images look similar to those in CIFAR-10 and CIFAR-100 datasets. I assumed it would be a fair assumption that the 32×32 resolution (as in these common benchmark datasets) should be enough to show some early results without much computational draw, at the cost of accuracy. More methods were eventually added as part of the Torchvision module in PyTorch (e.g. shears, color changes).

More importantly, image loading was handled as a NumPy array that needed to be converted into a PyTorch tensor, which requires rearrangement of dimensions. Also, some images were loaded with a single color channel (black and white images). These were handled by adding extra copies of the channel, so they could be processed in the same manner as color images.

5 Architecture Planning

Given the nature of the problem, the architecture of choice was a convolutional neural network (CNN). The many hyperparameters would be tweaked at different stages of prototyping, however, some functions would be fixed according to the task. The output layer will contain as many neurons as unique labels in the dataset. In order to have a probability distribution, the activation function will be set to a sigmoid function. The loss function was set to mean squared error (MSE) as it would handle a vector of probabilities adequately. Optimizer was set to stochastic gradient descent (SGD) with momentum unless otherwise specified. The batch size was preliminarily set to 4.

6 Architecture Tests

The first tested architecture in prototyping was a conventional highly simplified CNN. The aim was to test performance in the most basic but functional conditions. The next step was followed a similar approach in the opposite direction: a residual neural network model. A search of the current state of the art in CIFAR-10 classification (not multilabel, but similar image characteristics) showed good performance of ResNet. Consequently, an 8 block ResNet was trained and evaluated. Both attempts show very low performance, and the second had a very high computational cost. As a result, a raw scale-up of the sophistication of the model did not seem to overcome problems probably attributed to dataset imbalance.

7 Dataset and Architecture Issues

The main problem seems to be the dataset. There are labels wrongly assigned, and the data is quite imbalanced. The first strategy was to adjust the threshold to consider a positive label prediction. One proposed solution would be to adjust this threshold according to the distribution of labels: overrepresented labels were more likely to be predicted in general, and highly underrepresented labels may not even appear at all. Overrepresented label prediction should be penalized and underrepresented incentivized. However, there are other variables that need tweaking right now (model), so a simple heuristic was established for the threshold instead, as it seemed to work just good enough. At this point, the training loss with a simple unoptimized CNN could reach low enough values, but the evaluation showed poor results. The network was likely overfitting, so the was test loss was measured as well to confirm it. Only the six more represented labels were predicted as I could expect.

Data augmentation was adjusted to force random image greyscales as it could be possible that color would favor one of the overrepresented classes, but results worsened, only predicting the four most represented classes. Until now, images were also resized as a trade-off accuracy for faster training, but it might have been the moment to switch this simplification off. Training at full 128×128 was a really expensive calculation but improved results slightly. Further tests at a 64×64 showed that it could be a good trade-off between accuracy and training time. At the moment, training loss quickly converged to a minimum in one or a few epochs.

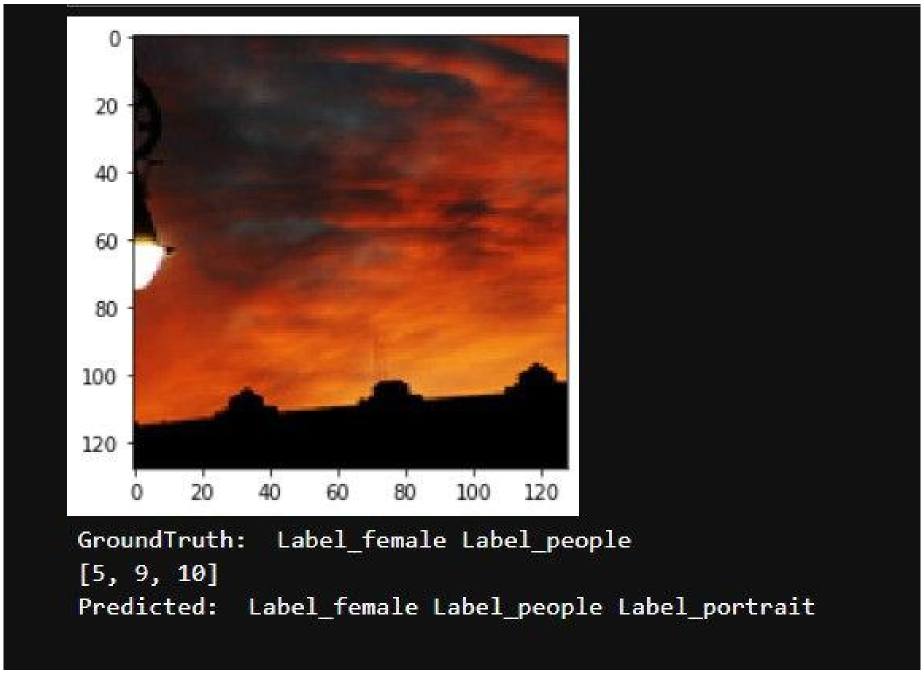
8 Architecture Optimization

It does not seem to be a full agreement of one strategy on how to design the network. Few online communities showed some agreement for certain issues, and experimentation still played the main role in architecture optimization.

Architectures with 3×3 convolutions seemed to be clearly favored, with higher convolution kernels for larger images. The performance was increased by a progressive shrinking in the convolution kernel, up to a 1 × 1 convolution the end for good training speed at the cost of some accuracy. About four convolutional layers seemed a good compromise.

To preserve accuracy as much as possible downsampling was used by max pooling. It provides a sort of horizontal and vertical translation invariance. A 2×2 kernel gave good results when the network had already learned something, so downsampling was applied to the later layers, and especially following a convolution. Max pooling at the beginning increased training time without much accuracy gain.

The batch size was also greatly increased, up to test with 64 samples.



```
GroundTruth:  Label_female Label_people
[5, 9, 10]
Predicted:  Label_female Label_people Label_portrait
```

Figure 1: Mislabeled image for the overrepresented class shows an apparent partially correct classification, which does not match reality

9 First Better Classifications

The proposed more optimized architecture with Adam optimization, and a random drop of samples labeled as “people” allowed the first better classification attempts. While dropping samples worked better than previous attempts, it produced overfitting. Other similar attempts dropping all combinations of “male”, “female”, and “people” resulted in overfitting. Other not so successful strategies included grouping the three previously mentioned labels as just “people”, but that increased the counts of the overrepresented label. Also, dropping “people” shifts the problem to the next overrepresented class.

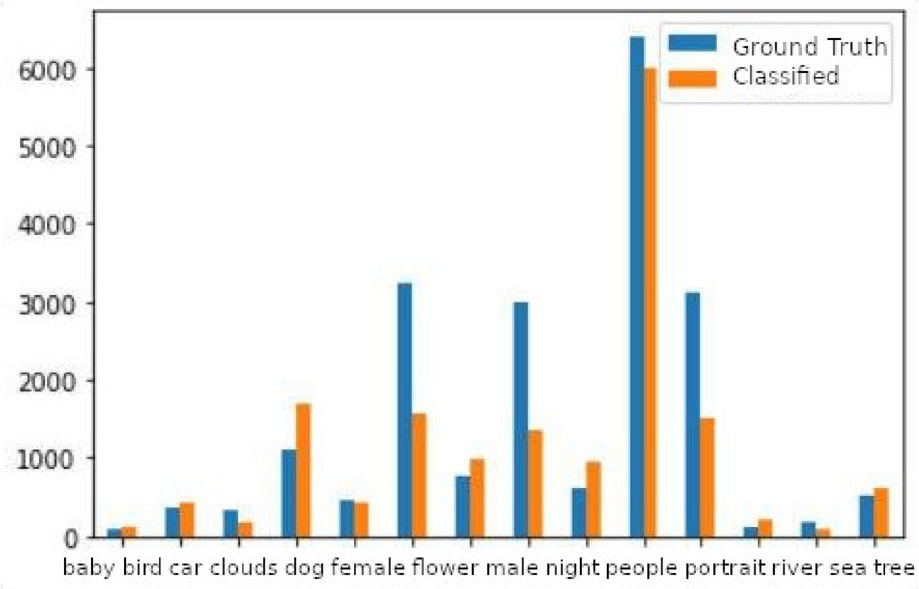
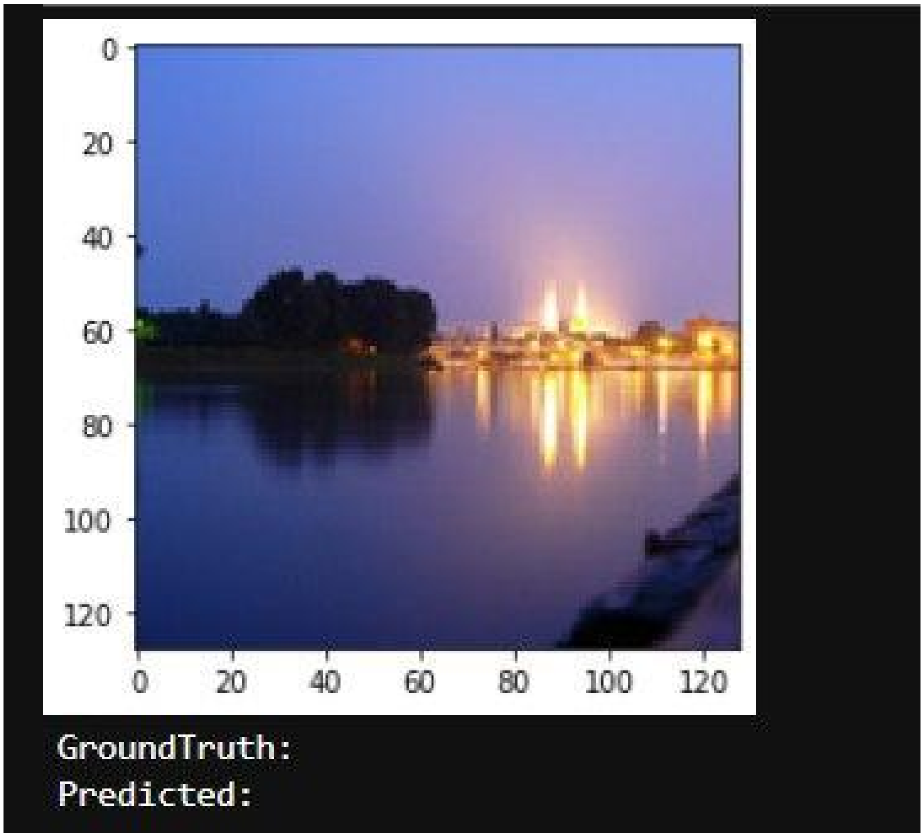


Figure 2: First better clasification attempt after optimizations, where the vertical axis indicates label counts. It does not include images without labels, but prediction imporved

10 Two Networks

Dropping all vastly overrepresented classes made a network that performed quite well. The task could potentially be split into a network that classifies everything that is not people, and another people classifier. Babies would be handled in a special way: it seems that an image with adult humans is always classified as “male” and/or “female” if body features allow doing so, and are otherwise classified as just people. It could be said that every time that the label “baby” appears, the label “people” should appear as well. The instances in which “baby” is not paired with “people” would be considered as an error in ground truth labeling, as there are already quite a few. This would allow classifying babies in this potential first network, and automatically append “people” to every “baby” classification.

Unfortunately, the second network never really worked, and some parallel attempts with a single network showed enough new potential, so the project continues in the single network direction.



```
GroundTruth:
Predicted:
```

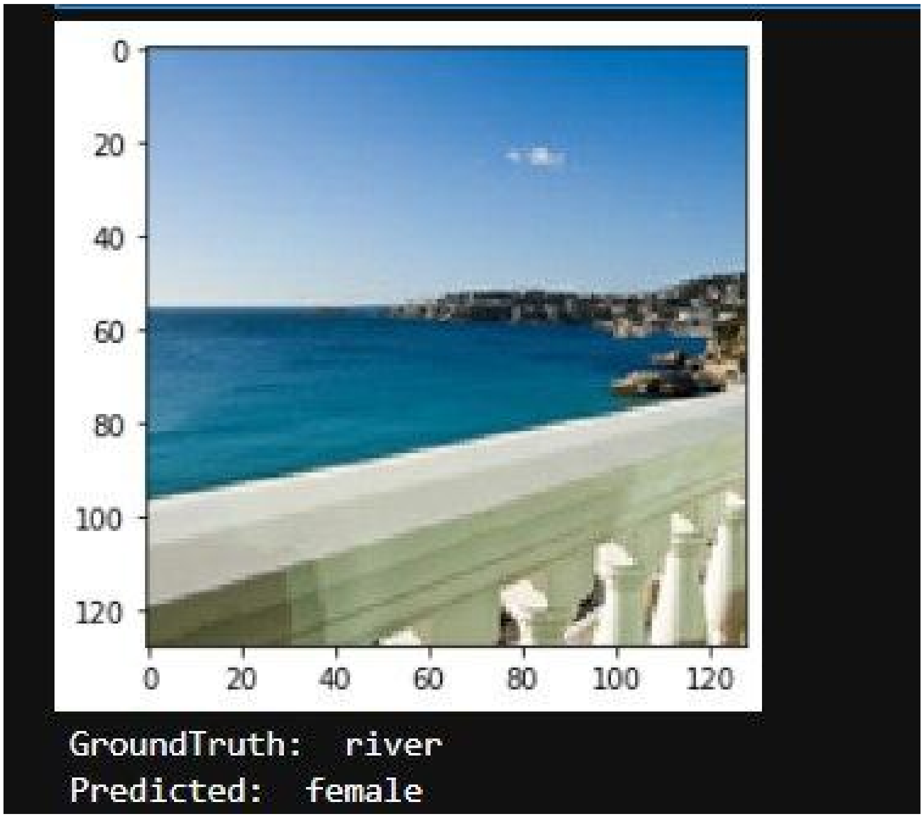
Figure 3: Empty labels correctly selected.

11 Too Many People

The overrepresented class seemed to be the main issue to solve for the classification to work properly. To correct the effect of the "people" layer, a method penalized false positives. Adjusting this value allowed for some fine-tuning of the false positives. The threshold was generalized for the other labels, outputs improved, but with room for improvement. In the end, it seemed like there should be some compromise with output accuracy as the dataset has many wrong labels.

12 Progressive Growing of CNNs

The network architecture had gone through many trials and error optimization attempts, replacing the loss function, adding dropout layers, etc. In addition, the first more rational design of the network involved 4 hidden layers. It was time to go simple again and start growing layers as long as accuracy and computation



```
GroundTruth:  river
Predicted:  female
```

Figure 4: The imbalanced dataset without a balancing method allows plenty of misclassifications like this one, a common case for the “people female male” set of labels.

time. The starting point was a single hidden layer, and then another one was added, only for the apparent Eureka moment. With a two hidden layer CNNs, plus balanced input data showed impressive results: a total of 98% of accuracy for all tested labels! It turns out that it was entirely due to sampling. Just by chance, all predicted labels just matched those in the sampled images for the test set. At the moment, it was plenty to think about: this could have happened at any point for false positives or false negatives. The apparent alternative was to perform cross-validation, but it would be a very expensive computation, especially for the number of trials I was launching. Cross-validation was then added and used more sparingly to increase the chances to catch these types of errors.

13 Testing Alternatives and Final Models

The performance seemed to have capped with balanced data, selecting the probability of each independent output neuron for a positive prediction, and having some sort of standard architecture. Data augmentation was also in place. Other trials included reducing the number of neurons and layers as the extremes of a few as possible, or as high as it allows to compute, testing as many variations of hyperparameters as possible. All kinds of activation functions were tested for the hidden layers. In many cases there was overfitting, the test and training split had to be adjusted, and data balancing revisited. In other instances, the network would never learn. With very few layers, abstraction was impaired, while many neurons made it not reasonable to compute. Despite this, the performance had seemingly reached a maximum difficult to improve for the better versions of the network and accessory methods, although initial training loss was still high due to dataset issues. In many attempts, the network assigned correct labels while the ground truth labels were incorrect.

Testing using accuracy is not correct especially in imbalanced data. For that, precision and recall are evaluated and taken into the F1 score metric.

Metric	*Value*
Precision	0.73
Recall	0.69
F1 Score	0.71

Table 1:

14 Conclusions

A convolutional neural network model can perform multilabel classification over a dataset of images. The performance was limited mainly due to data imbalance and errors in the ground truth. Some strategies can be used to overcome this problem, mainly:

- Balancing the input data.
- Adjusting the positive classification threshold.
- Optimizing network architecture.

Future work could include more sophisticated architectures and strategies for data balancing, augmentation, and ground truth error correction.

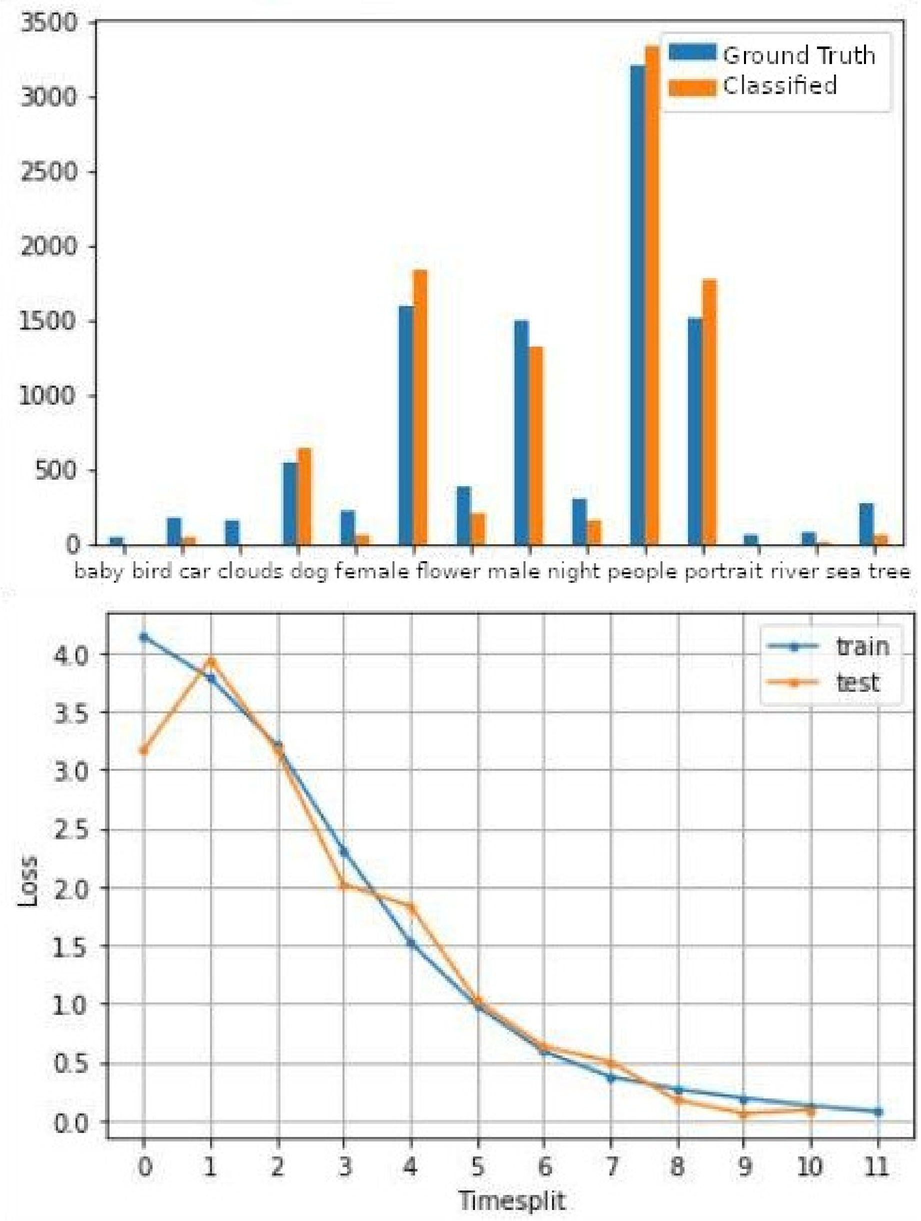


Figure 5: Some of the last models reached good performance, and lower signs of overfitting, but still with some margin to improve.