Presentation

Mathieu and Nastia

November 2022

First Iteration

Firstly, we built a simple CNN using Keras. It took two inputs: the data about the image (year, latitude, longitude), and the image itself. Then it processed the image data separately, applying convolutions and pooling, before flattening it and concatenating it with the metadata and adding Dense layers on top of that. The first model used Adam optimizer and minimized Sparse Categorical Crossentropy. The model was trained on a subset of training data for 10 epochs and showed the following results:

$$\text{Training loss: } 0.3261, \text{ Training accuracy: } 0.9144,$$

$$\text{Validation loss: } 1.2629, \text{Validation accuracy: } 0.4988.$$

So, it overfit quite dramatically to the training dataset. We compared different optimizers and got the following results:

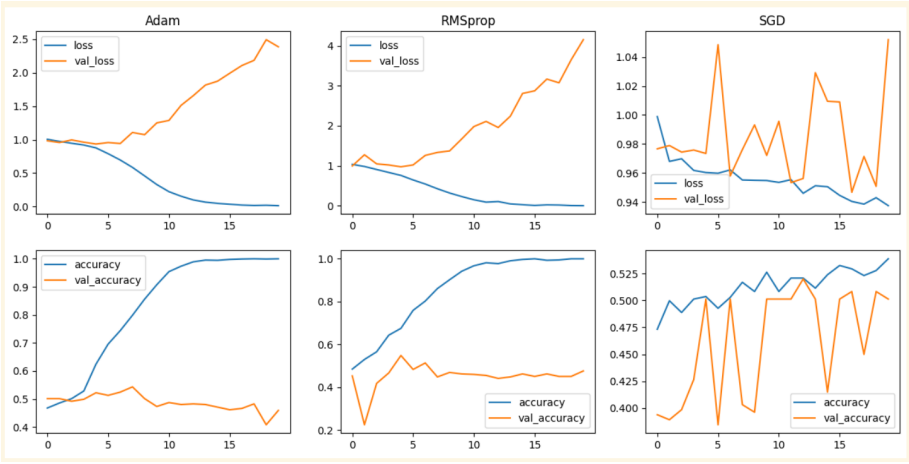

Fig.1: Performance of different optimizers.

Improvements

To improve our baseline model we decided to experiment with different model structures, as well as with image augmentation, as the aerial data contains the same features no matter how you rotate it. So we've defined a function that produces augmented data sets with images rotated 90 degrees and/or flipped along horizontal or vertical axis. As the classes present in the data-set turned out to be skewed (the 0 class was largely over-represented, while 1 class only comprised about 10% of the data-set) we also defined a class_weights array to tell the TensorFlow model, to use the weights when computing each gradient step, to favor the underrepresented classes.

It didn't work out as well as we'd hoped. The learning curves looked much like the ones in the Fig.1. So we turned to augmenting the data-set in such a way, as to balance out the classes. And still the model was overfitting. Only architectural changes could help us now.

With an added normalization layer and bigger steps in convolutional layers, to produce a model with fewer trainable parameters → hopefully less overfitting, we got this:

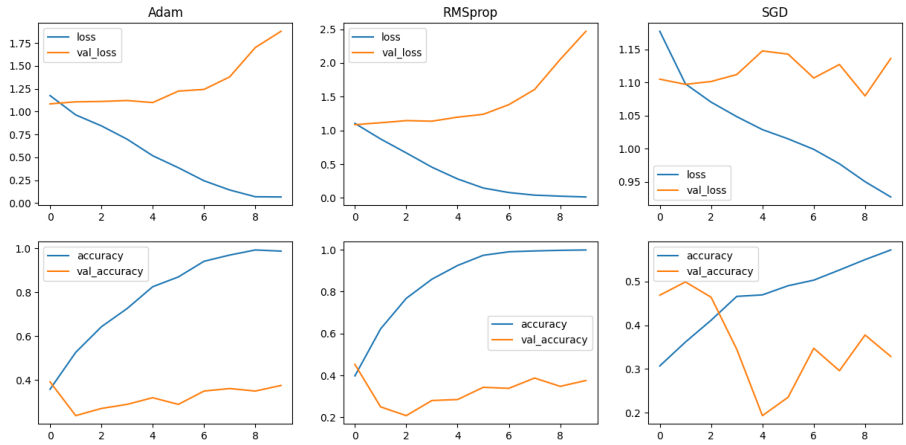

Fig.2: Performance of different optimizers.

Not much better, if any difference is to be seen at all. Unfortunately, though much more should have been done to produce a better model, time was actively working against us, and despite our best efforts, the power outage spanning 6 hours of this day took its toll.

In any case, this hackathon was a great learning experience, and we hope to learn even more from the feedback!