POLITECNICO
MILANO 1863

AN2DL - First Homework Report
LosPollosHermanos

Mohammadhossein Allahakbari, Michele Miotti, Francesco Pesce

mh2033, michelem, francescopesce

246639, 249499, 247974

November 23, 2024

1 Introduction

This report presents the results of the *image classification* task proposed in the first homework of the `Artificial Neural Networks and Deep Learning` course. The goal of the project is to classify blood cell images into 8 different classes by using *deep learning* techniques such as *Convolutional Neural Networks* (CNNs)[7].

2 Problem Analysis

The task involved the use of three datasets, only one of which was available to us. The others were used by the `Codabench`[11] platform to evaluate the developed models, with different datasets used in the Development and Final phases. The three datasets will be referred to as respectively `DS1`, `DS2` and `DS3`.

`DS1` consisted in around $12,000$ RGB images of blood cells, each with 96×96 pixels. We modeled the problem as a *multi-label classification* problem with 8 classes, where each image is an observation of a random variable with an unknown distribution.

After analyzing `DS1`, we found that it contained some outliers, i.e. duplicate and misleading images that were doctored by adding unrelated overlays, so we removed them from the dataset. Furthermore, after testing our first model on a local test set, we found that the model's performance on `DS2` was significantly lower. Since the mismatch appeared with multiple models, images in `DS2` could not be reasonably modeled as having been generated by the same distribution that generated the images in `DS1`, and were likely doctored in some way.

The samples of different classes in `DS1` were moderately imbalanced. However, using data augmentation to balance out the number of samples for each class seemed to indicate that the distribution of labels in `DS2` was very similar. As such, there was no need to adjust the prior distribution of the labels.

3 Method

Most of the development process was done on a `Jupyter` notebook, using the `Tensorflow`[1] and `Keras`[2] libraries for `Python`, mostly run locally using an `NVIDIA RTX 2060 Mobile` for training, due to strict GPU usage limits of popular platforms, such as `Google Colab`.

We initially split `DS1` into training, validation and test sets, but upon noticing that the accuracy on the local test set was unrelated to the accuracy on `DS2`, we decided to get rid of it, and apply a $80 - 20$ training-validation split.

All developed models contain a convolutional CNN core, due to two properties of CNNs: *automatic feature extraction* and *sparsity* of the weights with respect to dense neural networks, fundamental when the inputs are images. The *latent representa-*

tion is then used by a dense neural network, with 8 output neurons and a softmax activation. Since this is a classification problem, the loss function we employed was *categorical crossentropy*, as minimizing said loss function maximizes the likelihood of generating the data under our assumptions[9].

After experimenting with diverse techniques, as detailed in Section 3, we found that the best approach was to use a chain of two convolutional blocks, followed by two instances of a custom block, illustrated in Section 4, with downsampling after each block. The output is then fed into a `Global Average Pooling`[8] layer, used as a structural *regularizer*. The dense network at the top of the net has a hidden layer, followed by a `Dropout` layer for regularization, and `Batch Normalization`[5] is performed after each convolutional layer. The final model was trained using the `Adam`[6] optimizer with early stopping after 40 epochs of stagnation of validation accuracy, 10^{-3} initial learning rate and adaptively lowering the learning rate 5-fold after 20 epochs of stagnation in validation loss.

The final network has 1.9 million parameters, achieved an accuracy of 88.5% on `DS3`, and was developed after exploring a complex *design space* formed by the number of custom and convolutional blocks, presence of the hidden dense layer, optimizer used, learning rate and other variables.

4 Experiments

We performed many experiments, some of which significantly improved the accuracy of our models. In chronological order, the main breakthroughs were:

- **Simple CNN:** A simple CNN was developed to test the development flow and the platform, making us realize that `DS2` was likely doctored in some way.
- **Data Augmentation:** We implemented simple image augmentation techniques.
- **Overlaying:** After finding the outliers in the dataset, we hypothesized that the test set may have been doctored in the same way, so we experimented with applying overlays to the images in our dataset, using techniques such as convex combinations and multiplicative overlays.
- **Keras CV:** Using the `KerasCV` library[2] provided the first satisfactory results. The library provides richer augmentations, which can be applied in a complex pipeline. Since they are applied during training, augmentations can use different seeds at different epochs, fully utilizing the information in the training set. The overlaying technique was removed due to poor compatibility, and both the training and validation sets we augmented, since images in `DS2` were likely more "similar" to augmented ones.
- **Richer models:** We experimented with `Inception` blocks[10], `Residual networks`[4], `Global Average Pooling` and `Batch Normalization` layers.
- **Custom Block:** A custom block was implemented to mix the features of `Residual` and `Inception` blocks. The rationale and more details are discussed in Section 4.

We also performed experiments with less fortunate results. One that merits explicit mention is using *transfer learning*[12] from some well-known CNN architectures, such as `VGG19`, `ResNet50V2` and `MobileNetV2`, using the *ImageNet*[3] pre-trained weights, with a custom dense network. We attempted both training just the latter and unfreezing the top convolutional layers, but achieved very poor results with both techniques. These results, the large memory requirements of the models, problematic for both local training and the `Codabench` platform, and the wish to create our own custom model, led us to abandon this avenue. Other failed experiments include using the `Lion` and `SGD` with momentum optimiers provided by `Keras` in place of `Adam`, performing test-time augmentations, attempting to filter out noise in test images using Fourier transforms and filters, and mixed precision.

Other ideas were successful at first, such as averaging weights from all epochs below a certain validation loss as a regularization technique, and a voting mechanism with CNNs, acting as a sort of *ensemble* method. The ideas were effective on early models, but did not have an effect on more fine-tuned ones.

5 Custom Block

In 2015, `ResNet` revolutionized CNNs by injecting skip connections into the network, which help the learning process by making deeper levers learn *residuals* with respect to the identity. Adding them significantly improved our accuracy, up to 0.8 on `DS2`.

In `Inception` blocks, a set of differently sized filters are applied to the inputs, and the results are concatenated as output channels. Adding `Inception` blocks to our design did not improve the accuracy. However, contrary to previous experiments, were models classified correctly most of the same images as their predecessors plus some more, networks containing just simple `Residual` or just `Inception` blocks were able to correctly classify different images. This led us to hypothesize that the models captured *different features* of the dataset, and to combine the two into what, to the best of our knowledge, is a novel block. It is formed by stacking a number k of `Inception` blocks, and then inserting a skip connection, as shown in Figure 1.

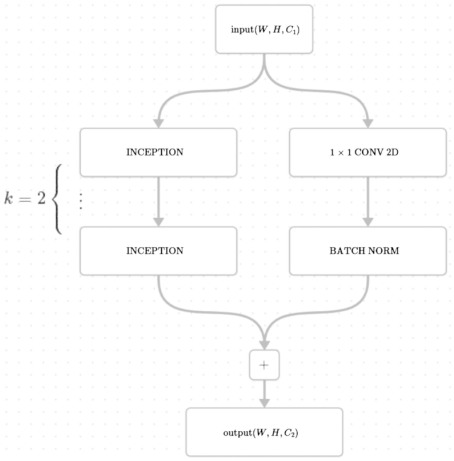

Figure 1: The structure of our custom block.

6 Results

Table 1: Results on `DS2` for each technique.

Experiment	Accuracy
Simple CNN	0.20*
Data Augmentation	0.25*
Overlaying	0.53
KerasCV	0.71
Richer models	0.80
Custom Block	0.88

During the Development phase, we trained and submitted more than 50 models. Table 1 shows the best result using each of the techniques discussed in Section 3, before introducing the next technique. Results marked with * are approximate, due to being obtained before an unexpected reset to submissions.

7 Discussion

The heavy usage of *augmentation* and *regularization* techniques allows the model to generalize well outside of the training set. While some larger models, such as ones obtained by *transfer learning*, may have been able to capture more features and achieve higher accuracies if trained well, our model can be trained on commodity hardware, has lower memory requirements and faster predictions. Moreover, we propose a *scaling* technique for devices with lower capabilites. By simply substituting one of the custom blocks with a convolutional layer, and reducing the number of channels in each block, a lighter model with less than 1/10-th of the parameters can be obtained, with a 0.82 accuracy on `DS2`. Using intermediate numbers of channels resulted in intermediate memory occupancies and accuracies on `DS2`.

8 Contributions

Most ideas and propositions were evenly distributed among the team members, while actual coding was more specific to each member's strengths. Many different branches were created in the project's repository to experiment with different ideas and techniques, and the final model was a result of combining the best experiments from each branch.

9 Conclusions

In this project, we developed and trained a custom CNN using techniques from the scientific literature. The model achieved an accuracy of 88.4% on the final test set, and makes use of a custom block we developed by reasoning about the rationale behind the blocks we were shown in class. Future work could include a more thorough analysis of the *design space* for the model, especially when it comes to the specific augmentation techniques, which seem to have a large effect on accuracy, and further attempts at *transfer learning*. Overall, the project's challenge-like nature and the lack of a pristine test set helped us follow a structured approach to solving the problem, giving us the opportunity to test out many different techniques.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] F. Chollet et al. Keras. `https://keras.io`, 2015.

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[4] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2015.

[5] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In F. Bach and D. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR.

[6] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[7] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel. Handwritten digit recognition with a back-propagation network. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann, 1989.

[8] M. Lin, Q. Chen, and S. Yan. Network in network. *CoRR*, abs/1312.4400, 2013.

[9] S. Shalev-Shwartz and S. Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[10] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions . In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, Los Alamitos, CA, USA, June 2015. IEEE Computer Society.

[11] Z. Xu, S. Escalera, A. Pavão, M. Richard, W.-W. Tu, Q. Yao, H. Zhao, and I. Guyon. Codabench: Flexible, easy-to-use, and reproducible meta-benchmark platform. *Patterns*, 3(7):100543, 2022.

[12] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 3320–3328, Cambridge, MA, USA, 2014. MIT Press.