Improve Lymphoma Subtype classification through Deep Learning with extensive hyper-parameter exploration

Cristiano Colpo[†], Filippo Dalla Zuanna[‡]

***Abstract*—Convolutional Neural Networks and general Deep Learning architectures are powerful tools for image classification; today this type of models can even surpass humans. Medical imaging is a field that can greatly take benefit from these new machine learning methods. However, life-threatening sectors cannot accept not well-defined solutions, so we cannot treat these new models as black-box. Instead, a well defined experimenting procedure needs to be followed and analyzed for achieving a satisfying result. In this paper we propose models that can achieve up to 98% of accuracy in classify lymphoma subtypes, highlighting that we need to focus on obtaining an explainable model instead of throwing a large and complex one that we cannot fully understand. Only in this way, these powerful methods can express their enormous potential.**

***Index Terms*—Medical Imaging, Supervised Learning, Optimization, Neural Networks, Convolutional Neural Networks**

I. Introduction

Thanks to the democratization of computing power introduced by free cloud computing services, researchers all over the world are experimenting with complex neural network models to resolve all kinds of problems. Medical imaging is one of the sectors where these new methods can, in theory, yield better results than the classics ones. In practice, the medical field, like other life-threatening sectors, requires solutions that have to explain results with a certain degree of accuracy. Deep Neural Networks behaviour can be complex to explain, and with the increase of complexity, this problem can only aggravate. As soon as invented in the 80s, it was clear that Convolutional Neural Network (**CNN**) architecture would be perfect for image recognition and classification. Almost 30 years later with AlexNet [1], finally apply deep networks for tasks related to images gave successful results.

In this paper we want to illustrate how, handling the work of some previous research [2] [3] in a different way, the performances of state of the art algorithm for classification of lymphoma subtypes can increase. Worldwide, these types of cancer developed in 566,000 people in 2012 and caused 305,000 deaths [4]. But the specialized Doctors needed for the diagnosis of this disease are never enough, so with the aid of Machine Learning (**ML**) we want to prove that new computational models can support the work of our doctors. Moreover, we criticize some of the decisions taken by the cited papers about testing and training methods. This because a precise and clear procedure is the starting point for the clarity of the solution.

[†]email: {cristiano.colpo}@studenti.unipd.it
[‡]email: {filippo.dallazuanna}@studenti.unipd.it

II. Related Work

Our benchmarks for this work are the image classification method proposed in Orlov et al.(WND-CHARM) [5], the AlexNet model build by Janowczyk et al. [3] and the model based on InceptionV3 of Tambe et al. [2].

The classical method of image classification WND-CHARM [6] is a multi-purpose image classification using a compound image transforms. This algorithm extracts 1025 features for each image, including polynomial decompositions, high contrast features, pixel statistics, and textures. These features are computed on the raw image, transforms of the image, and transforms of the transformed image. It is not possible to use this so large set of features. Moreover, a large portion of them is noise, so they decided to assign a weight to each feature based on the Fisher Discriminant score. The resulting set with the reduced and weighted features is used in a variation of nearest neighbour classification that they named Weighted Neighbor Distances (WND-5). With this method, they calculate the feature vector of the test image and the distance of such a vector to the vector of the class. The results of this proposed algorithm applied to our dataset can be found in the NIA site [7], with 85% of accuracy. This rather old machine learning algorithm is really important, even if it has not the best accuracy. This is because well-known processes and mathematical proofs define the output; instead, the other two methods at the base of our work are more like a "black box".

In "Janowczyk et al." the purpose of the paper is to show the flexibility of Neural Networks at resolving image classification tasks, hence they use a discretionary AlexNet configuration to solve various problems in the medical images field, achieving 96% of accuracy in the lymphoma subtype problem with a majority vote rule prediction over the patches of the test images.

In "Tambe et al." they used the InceptionV3 architecture for improving the results of Janowczyk's research in lymphoma subtype classification, by performing a hyperparameter optimization on their model instead. With this new and complex model introduced by Szegedy et al. [8] in 2016, they obtain better results with 97.33% of accuracy, training and testing over the resized images.

III. Processing Pipeline

The entire processing pipeline was studied for performing our analysis to obtain results in the less amount of time, given a relatively small computing power. We wanted to perform all the operations inside an environment that could be replicated everywhere. Hence all the operations are performed with Python. The dataset [7] used was divided into training, validation and test set with the python library `Split-folders` with 0.8, 0.1 and 0.1 ratios respectively, wherein each subset we will find the three directories of each class of images.

The way we feed the network was studied to reduce to the minimum training time for each epoch. We started using the Keras classes to take advantage of its augmentation methods, but we realized that they weren't optimized for our task. For this reason, we waived the additional advantages of these methods switching to a more performing API, the `tf.data` and the `TFRecords` format for training our deep neural networks. The result is an x20 decrease in time concerning the slow and non-optimized high-level framework Keras. The TensorFlow Records files are created with a procedure where each image was decomposed into patches with a given size. All of these were serialized into the corresponding dataset file(training, validation or test). In this way, each file contains all the patches of all the images belonging to the training, validation or test division, in a binary lossless format.

This backbone of our processing pipeline is the starting point of our work, an efficient data structure that can be managed for different data elaboration functions like shuffling, cropping or transformations. These functions are managed by low-level APIs, including the relative memory allocation(a crucial aspect in big datasets). These are not trivial and need lots of studies to be mastered, so we rely on the default configuration since it seems to work well and our searches regarding this aspect were not so useful. Then, after the decision of the parameter configuration schedule, the dataset was consumed for various tasks by different deep networks structures that have been implemented in this project, like the one in figure 1.

IV. Images

The dataset obtained from NIA [7] is composed of 374 medical images taken from biopsies, having dimensions 1388x1040 and with 3 channels(RGB images). The particular colouration came from the colouration with Hematoxylin and Eosin; this permits to see the image at the microscope. The images stand into three categories of malignant lymphoma, cancer affecting the lymph nodes:

- CLL (chronic lymphocytic leukaemia)
- FL (follicular lymphoma)
- MCL (mantle cell lymphoma)

The dataset presented is a collection of samples prepared by different pathologists at different sites. There is a large degree of staining variation that one would normally expect from such samples. It is common to use lossless format for medical images because sometimes lossy compression can invalidate a diagnosis [9]. We wanted to follow this approach, so we didn't downsample or convert the images into another format.

A. Dataset creation

For training the models, we created TFRecords files by decomposing each image into patches of dimension decided according to the task we wanted to achieve, challenging some of the states of art models such as AlexNet and InceptionV3. Regarding the first model, the patches have dimension 36x36 pixels. Instead, the second one has 347x260 pixels. Following the method used in existing literature [3], we performed a random crop for each patch by the right input dimension of the neural network of reference, so 32x32 for AlexNet model and 224x224(an arbitrary number) for InceptionV3. This extraction is followed by a random rotation of plus or minus $\frac{\pi}{2}$ with probability 0.5, only for the training set. For the test set, we performed 5 random crops for each 347x260 patches only, instead of 1 for the training set, the validation set and the 36x36 configuration of the test set. Regarding the training set randomity, the images have been mixed during the patches serialization. However, these patches results for each image in order, hence we decided to shuffle the entire set, after crop and eventually rotation. In this way, each input is extracted randomly from the next 10000 samples. Each one is so originated from different images as much as possible. Our resulting dataset beehive like a generator, so it will regenerate every time it has consumed all his items in the training steps. This process is all automated in the underlying API of Tensorflow.

V. Learning Framework

A. Alexnet model

Starting from the work done in [3] paper, we replicate the setup using the standard structure of Alexnet Deep Learning model with the extracted image patches of 36x36x3 and the input patches of 32x32x3, as described in section III. Then, instead of training the model with an arbitrary configuration, we proceed searching parameters with a validation process; in this manner, we can choose the configuration that fits the problem better. We search through three types of optimizers commonly used in Deep Learning problems, Adagrad as in the reference work, stochastic gradient descend (**SGD**) with Nesterov momentum and Adam. Using these, we perform hyper-parameter optimization w.r.t. the learning rate, comparing 0.01, 0.001, and 0.0001 values, and the batch size, 64, 128, and 256 values, parameters responsible for the stability of the learning and generalization of the training. In total we validate 15 configurations, including the one of the reference work, selecting the learning rate first(using 128 as starting batch size parameter) and then the batch size, depending on the results of 8 epochs with 5000 batches per epoch training. The evaluation and so the selection is based on the test accuracy with more importance(3 times more) and the test loss. This kind of training is motivated because the overall budget used results in 600000 (15x8x5000) batches processed, the same amount of data used for the training of the original model, so,

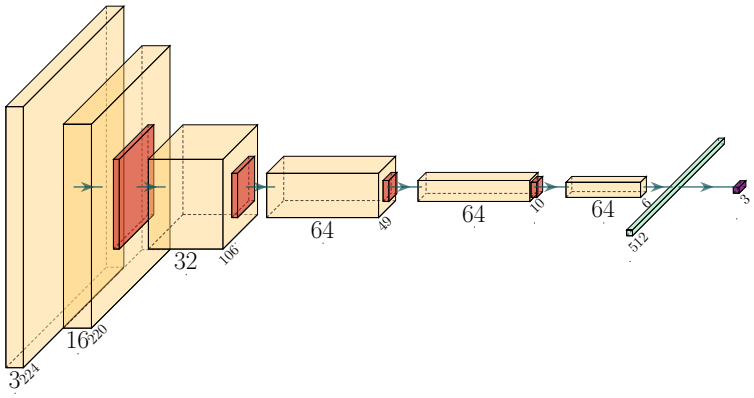

Fig. 1: An example of one of the CNN configurations

in this way, the effort searching the parameters is acceptable. Finally, we train the best configuration found and the reference work's one for 120 epochs with 5000 batches per epoch, using the same budget as before.

B. InceptionV3 model

This part of the research use as reference the work done in [2] paper. In this case, we compare the performance of the same InceptionV3 model using patches instead of the complete resized images; we wanted to see if resizing images are compromising information and the effectiveness of the training. Then we search for a simpler but effective customized model(in the next section) since the InceptionV3 model is very complex and requires a lot of computational resources and training time. For these models, we use as data the extracted image patches of 347x260x3 and the input patches of 224x224x3, as described in section III. Regarding the first part, the information on the reference work parameters is inaccurate(the optimizer is not provided). Then, we provide some attempts for comparison. This is done by training the model with all three optimizers used for Alexnet (Adagrad, Adam, SGD) with the same parameters as the reference training procedure:

- 35 epochs
- 20 batches for epochs
- 32 as batch size
- 0.001 as learning rate

An ad-hoc hyper-parameter selection is done to improve the unexpected results that will be discussed later. Then, training is performed with all the three optimizers, for 35 epochs, with 200 batches per epoch, 64 as batch size, and 0.0001 as learning rate.

C. Our CNN5 model

We developed a more lightweight solution starting from a basic CNN model with 5 convolutional layers. We explored this basic configuration at first, then we tried to arrange the structural components in a way to increase the performance of the overall model. The training parameters are the combination of 0.001 and 0.0001 learning rates with 64 and 128 batch sizes, with the usual 3 optimizers(total of 12 models). Then, as structural components, we change the initial fully connected section(one layer of 512 neurons) to 256, 1024 and adding another layer(two-layer of 512). Also, we modify the number of convolutional layers to 10, the dropout parameter to 0.5, the activation function to Leaky ReLU, and the input patch size to 168x168x3. All the CNN models have a training procedure of 30 epochs, with 500 batches per epoch referred to the batch size of 128, so 1000 batches in case of 64 as batch size. In summary we tweaked this hyper-parameters:

- learning rates: 0.001, 0.0001
- batch sizes: 64, 128
- convolutional layers: 5, 10
- activation functions: ReLU, Leaky ReLU
- fully connected layers: 256, 512, 1024, [512, 512]

D. Methods of testing

According to the method we use to test our models, results can vary to a certain degree. The winner-take-all voting method, used in the paper of Janowczyk et al [3], defines the class of an image based on the most frequent class assigned to the patches of the said image. This method is biased because it assumes that global features decide if an image belongs to a class. Hence it is something presented in the entire, or most of, the picture. Moreover, this strong assumption can be invalidated, if the image didn't contain only the lymph nodes, (e.g. the model is not robust to occlusion). Fortunately, looking at the result of the experiments that used the entire image downsampled, it seems that these features are indeed global, anyway, we decided to use a more general approach testing our models by stating the accuracy as per patch.

Remark V.1. All the training is performed in the Google Colab platform mostly with the available Tesla T4 16GB GPU and some cases with Tesla P100 16GB GPU that have, anyway, comparable performance.

VI. Results

A. Alexnet model

Regarding the work done for the Alexnet model, we state that the parameters validation procedure effectively improve the selection of a better model since the best model found reaches a remarkable performance gain, with 80% accuracy instead of 66% of the related work configuration. The latter differs from the original results since the test evaluation is not a majority vote rule but an evaluation over general test patches. The chosen learning rate is 0.001 because it performs better than the others in almost all the optimizers, only in Adagrad it performs similarly to 0.01, but the training history shows that the accuracy is not increasing (figure 6). Then, over the batch sizes comparison, the selected configuration results in the SGD optimizer with 128 as batch size. In this case, the test error is the discriminant of the selection, since that some configurations have pretty the same accuracy.

The final training history (figure 3) indicates why that model was selected and why it improves the performance compared to the reference configuration (figure 2). It shows that the validation accuracy increases more and more rapidly at the beginning, denoting that it's learning faster. Also, it maintains and refines what learned since the accuracy doesn't deteriorate, generalizing as well.

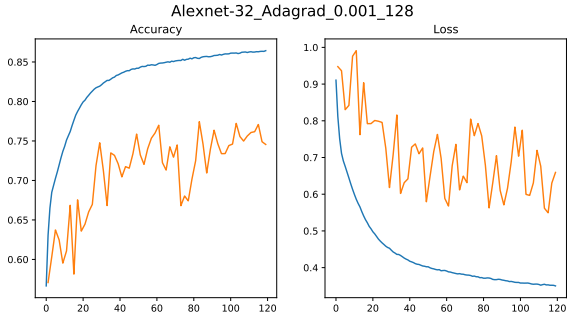

Fig. 2: Reference AlexNet training

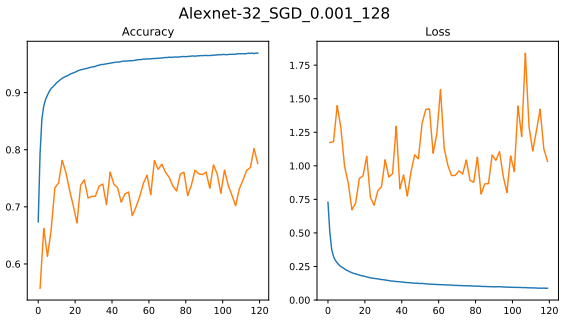

Fig. 3: Best AlexNet training

B. InceptionV3 and Our CNN model

Starting from the InceptionV3 comparison with patches, we don't provide any meaningful results. This is caused by the setup described in the related work that, unless there are misunderstandings, seems to be inadequate and incomplete. The training procedure (figure 10) is fast(less than a minute), poor(few samples processed), clearly unstable, without an optimizer configuration and with accuracy(about 90%) far from the target one(97%). For this reason, we change and expand the training set up to get a configuration that works. We increase the batches processed per epoch from 20 to 200 to include more training samples and we set the learning rate to 0.0001 and the batch size to 64 to improve stability. With this configuration, the Adam optimizer can reach 98% accuracy, which is a great result even if the training history (figure 11) is not so meaningful.

TABLE 1: AlexNet final model comparison

optimizer	l.r.	batch size	accuracy(%)	loss
Adagrad	0.001	128	67.79	0.7079
SGD	0.001	128	80.61	0.7594

Then we give up the comparison between image patches and image resize methods since it wasn't our intention to find a configuration that fits well the resize method; this is because the latter cuts indiscriminately the input information and this is not recommended. Instead, we focusing on challenging the InceptionV3 configuration found, since we suggest that a general state of the art model does not necessarily do the best in every problem only because of its complicated and well-studied structure. We find that the best configuration for our customized CNN model, reaches a comparable accuracy of 94%, with half of the training time and a more lightweight conformation. This means that it isn't necessary to rely on a complex state of the art model to solve every problem. The parameter exploration process leads to the initial structure with Adam, 0.0001 learning rate, and 128 batch size. All the other attempts done to improve the model doesn't reach significantly better performance. The increase of the dropout parameter doesn't lead to a better generalization cause the original parameter is enough. Changing or complicating the network with more convolutional layers or neurons doesn't permit to learn the problem better. The same result came up with the change of the activation function or the input dimension. Even doubling the training time doesn't increase the performance so much. This means that in this configuration it is not possible to reach notable improvements.

VII. Concluding Remarks

And there we are, we set up a procedure to efficiently manage the NIA [7] dataset to train some neural network models that can solve the lymphoma subtypes classification task. This succeeds with high accuracy using image patches with a customized model, procedure inspired by two other setups and finalized expanding their research. First of all, to justify and permits the replica of the results, we want to highlight that a clear and meticulous program is needed. In this way, it is possible to investigate this kind of research over and over. That starts from the hyper-parameters schedule since a parameter exploration procedure is needed, in deep learning

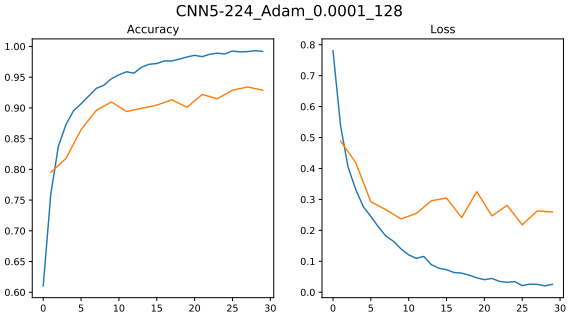

Fig. 4: Best CNN5 training

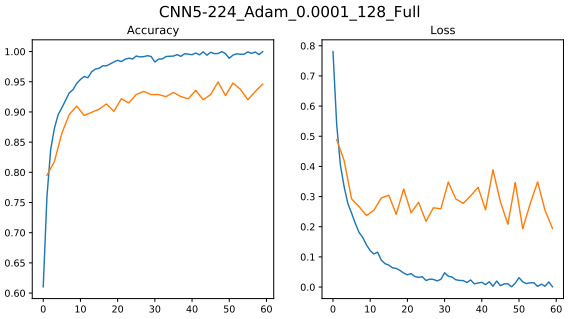

Fig. 5: Best CNN5 with double training

problems, to find the configuration that fits the most, as shown in a part of our work. Usually, handcrafted parameters don't lead to a perfect configuration and sometimes are even difficult to be motivated. Another relevant aspect is the abuse of the state of the art models for all kinds of problems; these are very complicated and resource-consuming so the better performances are not worth it in some cases. With our configuration, indeed, we reach a comparable performance in half of the training time and with a simpler model that is managed with less memory space required. If the precision needed would be more stringent than the results, certainly the necessity of bother more complex models occurs. But in this case, there wasn't such target to reach.

TABLE 2: CNN5 learning rate and batch size comparison

optimizer	learn rate	batch size	accuracy	loss
Adagrad	0.001	64	82.50	0.4291
	0.001	128	76.41	0.5888
	0.0001	64	59.53	0.8843
	0.0001	128	48.75	0.9700
SGD	0.001	64	91.72	0.2905
	0.001	128	89.53	0.3207
	0.0001	64	81.72	0.4254
	0.0001	128	76.72	0.6040
Adam	0.001	64	90.00	1.3977
	0.001	128	87.81	0.7617
	0.0001	64	94.69	0.3046
	0.0001	**128**	**94.38**	**0.2268**

TABLE 3: CNN5 structural components comparison

structural parameter	accuracy	loss
original	**94.38**	**0.2268**
256 neurons	89.53	0.3380
1024 neurons	90.47	0.4615
2 layers of 512	94.22	0.2451
10 conv. layers	92.97	0.7599
0.5 Dropout	91.56	0.3682
Leaky ReLU	92.81	0.2680
168x168 input	91.41	0.3270
double traning	94.38	0.2540

These results need a context to be practically used and this is also another problem for this kind of research. The output of these models is a prediction that is justified only by the presence in the input of some particular abstract features. These have been encountered during the training procedure. These, for a complete utilization, need to be extracted, showed and validated by a professional doctor, in a way that the prediction has a counterpart motivation on the traditional studies. This problem opens the doors to a series of researches about the interpretability of the deep learning results. In particular with medical images, that are necessary for the complete utilization of our models. Only in this circumstance, the presented models could be used to divide the more probable cases to the less probable in a way to accelerate and facilitate the doctor's analysis. The stand-alone use is very far from reality since the sensitive implications of the problem and the need for justification, someone's responsibility and ethical consent in the actual society. Who will accept to have diagnosed a serious disease by an algorithm with no justification and with nobody taking responsibility? The road is long, but this kind of research is a starting point to a cheaper, more accurate and more accessible health care.

A. *Lessons learned*

This project was challenging, it has made us understand how difficult is to reproduce the results of existing papers. We think that without a common environment, we cannot resolve complex problems. So we decided to make the code available to everyone. Working with Tensorflow made us understand how powerful this framework can be. But also we have been aware of its limitations, for example, the data management thing. Then also we have learned better how a clear procedure is needed to manage Deep Learning researches. This is because explaining some behaviours was very difficult and sometimes impossible. For this reason, the planning of the work is the only way to be sure to explain results, with numbers and procedures, right or wrong they are. Trying different parameters regarding a personal idea or behaviour can lead to unexpected results that complicate further the work. That lesson gives us an improved capability to manage this kind of problems, also addressed in some other courses.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[2] R. Tambe, S. Mahajan, U. Shah, M. Agrawal, and B. Garware, "Towards designing an automated classification of lymphoma subtypes using deep neural networks," in *Proceedings of the ACM India Joint International Conference on Data Science and Management of Data*, pp. 143–149, 2019.

[3] A. Janowczyk and A. Madabhushi, "Deep learning for digital pathology image analysis: A comprehensive tutorial with selected use cases," *Journal of pathology informatics*, vol. 7, 2016.

[4] W. H. Organization, *World Cancer Report 2014*. 2014.

[5] N. V. Orlov, W. W. Chen, D. M. Eckley, T. J. Macura, L. Shamir, E. S. Jaffe, and I. G. Goldberg, "Automatic classification of lymphoma images with transform-based global features," *IEEE Transactions on Information Technology in Biomedicine*, vol. 14, no. 4, pp. 1003–1013, 2010.

[6] N. Orlov, L. Shamir, T. Macura, J. Johnston, D. M. Eckley, and I. G. Goldberg, "Wnd-charm: Multi-purpose image classification using compound image transforms," *Pattern recognition letters*, vol. 29, no. 11, pp. 1684–1693, 2008.

[7] "National institute on aging, intramural research program laboratory of genetics." https://ome.grc.nia.nih.gov/iicbu2008/lymphoma/index.html. Accessed: 2021-01-21.

[8] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

[9] E. S. of Radiology (ESR *et al.*, "Usability of irreversible image compression in radiological imaging. a position paper by the european society of radiology (esr)," 2011.

Appendix

TABLE 4: AlexNet learning rate comparison

optimizer	learning rate	accuracy(%)	loss
Adagrad	0.01	69.24	0.7636
	0.001	63.45	0.8463
	0.0001	48.76	0.9955
SGD	0.01	39.18	1.3695
	0.001	**70.07**	**0.7826**
	0.0001	64.17	0.7823
Adam	0.01	32.50	1.1648
	0.001	71.85	0.9184
	0.0001	65.63	0.8777

TABLE 5: AlexNet batch size comparison

optimizer	batch size	accuracy(%)	loss
Adagrad	64	66.54	0.7908
	128	63.44	0.8477
	256	57.24	0.9127
SGD	64	70.59	0.8915
	128	**69.96**	**0.7828**
	256	67.77	0.9222
Adam	64	37.50	1.0966
	128	71.88	0.9169
	256	67.14	0.8763

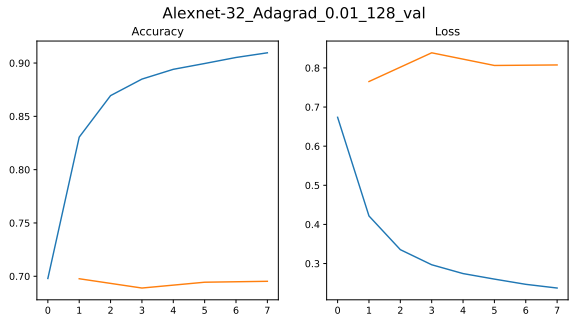

Fig. 6: AlexNet Adagrad 0.01 learning rate training

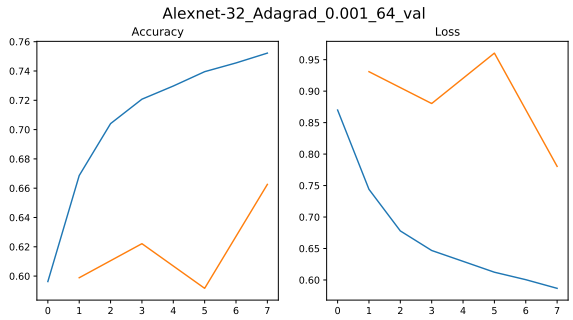

Fig. 7: AlexNet Adagrad 64 batch size training

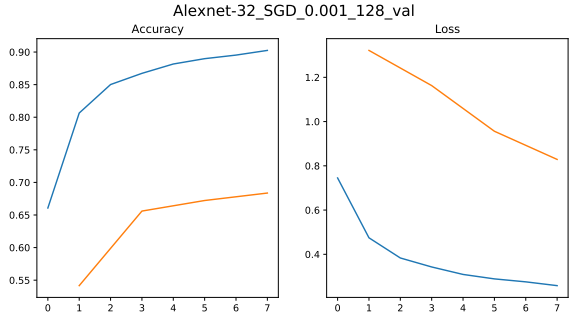

Fig. 8: AlexNet SGD training

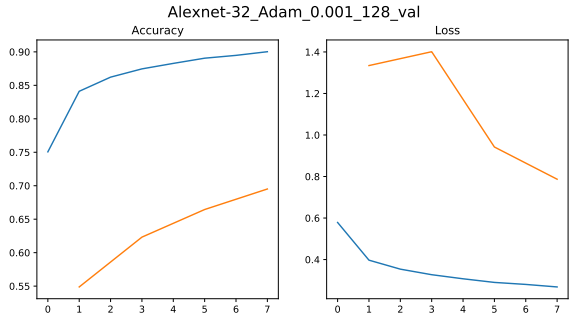

Fig. 9: AlexNet Adam training

TABLE 6: InceptionV3 results

	reference config.		improved config.	
optimizer	accuracy	loss	accuracy	loss
Adagrad	90.78	0.2600	87.34	0.3269
SGD	93.13	0.1803	93.28	0.2474
Adam	84.22	0.4165	**98.12**	**0.1288**

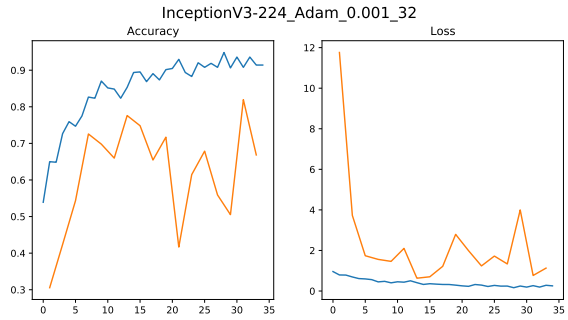

Fig. 10: InceptionV3 bad Adam training

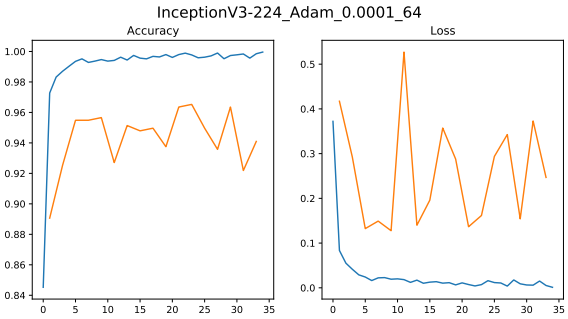

Fig. 11: InceptionV3 improved Adam training