Computer Methods and Programs in Biomedicine (2023)

Contents lists available at ScienceDirect

Computer Methods and Programs in Biomedicine

journal homepage: www.elsevier.com/locate/media

# Evaluation of deep learning training strategies for the classification of bone marrow cell images

Stefan Glüge[a,*], Stefan Balabanov[b], Viktor Hendrik Koelzer[c], Thomas Ott[a]

[a] *Institute of Computational Life Sciences, Zurich University of Applied Sciences, Schloss 1, 8820 Wädenswil, Switzerland*
[b] *Department of Medical Oncology and Haematology, University Hospital Zurich and University of Zurich, Rämistrasse 100, 8091 Zurich, Switzerland*
[c] *Department of Pathology and Molecular Pathology, University Hospital Zurich and University of Zurich, Schmelzbergstrasse 12, 8091 Zurich, Switzerland*

ARTICLE INFO

*Article history*:

ABSTRACT

## 1. Supplementary Material

This supplemental material file provides additional information about the study, including details on the evaluation metrics, model architectures, random initialization methods, learning rate decay strategies, and the Grad-CAM technique.

### *1.1. Evaluation metrics*

For quantitative evaluation, we use the measures of precision, recall and F1 score. Their definition is based on the number of images classified true positive/negative (TP/TN) and false positive/negative (FP/FN), respectively. True positive/negative are the number of images classified or not classified, respectively, into a given class in agreement with the ground truth. Similarly, false positive/negative signify the number of images classified or not classified, respectively, into a given class in disagreement with the ground truth.

Precision computes as

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{1}$$

and recall

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{2}$$

Hence, precision reports what proportion of positive identifications by the model are actually correct, and recall reports what proportion of actual positives are identified correctly by the model.

*Corresponding author: stefan.gluege@zhaw.ch

Precision and recall have a trade-off and cannot both be increased at the same time. Depending on the problem at hand, one has to decide to focus on reducing false positives (increase precision) or reduce the false negatives (increase recall).

The F1 score is basically the harmonic mean of precision and recall

$$\text{F1 score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}, \tag{3}$$

and thus, reports a score that balances precision vs. recall.

Table 4 shows the mean precision, recall and F1 scores that were obtained in the 5-fold cross-validation of the different models under different pre-training conditions. These scores were calculated as follows: (i) compute the class-wise score for each of the 21 classes for each of the five folds; (ii) compute the mean ± standard deviation of these class-wise scores over the five folds; (iii) compute the mean over all classes, resulting in a single score for each cross-validation experiment. Note that all classes are weighted equally, regardless of how many samples they contain.

The F1 score is a good metric to use when the cost of FP and FN are equal. However, when the cost of FP or FN is different, then the Matthews correlation coefficient (MCC) introduced by Matthews (1975) is a better metric to use (Boughorbel et al., 2017). In our case, we don't know the cost of a mispredicted cell image, so we chose to use the F1 score, implicitly assuming that the costs for FP and FN are equal.

When calculating the F1 score for each class, it is important to note that the majority class was labeled as negative. This is due to the convention that rarer or more 'interesting' samples are usually labeled as positive. For these problems, the F1 score serves its purpose as a good metric by giving more weight to the positive class.

To access the difference between MCC and F1 score, we computed them for all four models for the pre-training scenario ImageNet + CD, which yielded the best models. The differences between the MCC and F1 scores are negligible. Thus, the F1 score, in this case, combines precision and recall in a more interpretable way than MCC does.

### *1.2. Model architectures*

In this section, we introduce the general model architectures used in our study, starting with the general concept of Convolutional Neural Networks (CNNs) in the image classification task. From there, we describe the models in more detail in the order of their publication.

A classical CNN architecture for image classification consists of a stack of convolutional layers followed by pooling layers and fully connected layers. The convolutional layers extract features from the input image, while the pooling layers reduce the spatial dimensions of the feature maps. The fully connected layers learn to classify the extracted features into different classes (LeCun, 2012).

*VGG*, proposed by Simonyan and Zisserman (2015), is a classic CNN architecture that uses a stack of small convolutional filters. VGG-19 consists of 19 layers, including 16 convolutional layers and 3 fully connected layers. It is one of the deepest CNN architectures proposed at that time.

*ResNet* (He et al., 2016) is a CNN architecture that uses residual connections. In a classical CNN, each layer learns to extract features from the output of the previous layer. As the network gets deeper, it becomes more difficult to learn these features because the gradients of the loss function can become very small as they propagate through the network. Residual networks solve this problem by using a skip connection between every two layers, along with the direct connections between all layers. A block of such a connection is called a 'residual block', and these are stacked on top of each other in a ResNet to maintain efficient learning of parameters from the identity function, even in deeper layers. ResNet-152 has 152 layers, including 150 convolutional layers and 2 fully connected layers.

*RegNet* was proposed by Radosavovic et al. (2020) with the goal of designing a flexible network architecture. It should be efficient to run on mobile devices, but also be accurate when adapted for the best classification performance. This adaptation is controlled by setting the parameters in a quantized linear function to determine the width and depth of the network. Ultimately, RegNet is not an architecture, but a network design space.

A network design space is a set of different parameters that define a space of possible model architectures. Such parameters can be the width, depth, groups, etc. of the network. First, the authors define a space of all possible models which they call AnyNet (all kinds of models from all kinds of combinations of the different parameters). All possible models are trained and evaluated on the ImageNet dataset. From this AnyNet space, they created simplified versions of the AnyNet design space by analyzing which parameters are responsible for the good performance of the best models. Improvements from a general design space to a narrower design space include setting a shared bottleneck ratio and a shared group width, and parameterizing the width and depth to increase with later stages. Finally, the optimized RegNet design space contains only good models and also the quantize linear function necessary to define the models.

In the end, the RegNet design space is composed of multiple stages consisting of multiple blocks that form a stem, body and head. Multiple stages form the body. Each stage is composed of multiple standard residual bottleneck blocks with group convolution (Sick, 2021). The RegNet_y_32gf network used in our comparative study is optimized for classification accuracy (top models from the RegNetY design space) within the 32 billion floating point operation (FLOP) regime (Radosavovic et al., 2020).

*ViTs* (Dosovitskiy et al., 2021) are based on the transformer architecture, which was originally developed for natural language processing tasks (Vaswani et al., 2017). Unlike classical CNN architectures, ViTs divide the input image into patches and then use a transformer encoder to extract features from the patches. The transformer encoder is a self-attention mechanism that allows the network to explicitly learn long-range dependencies in the image. This is the main difference to CNNs, which learn long-range dependencies implicitly, i.e. they learn to detect long-range dependencies in the image by stacking convolutional layers together. However, this mechanism makes ViTs more computationally expensive to train and deploy. The ViT_l_32 model is based on the 'large' configuration used for BERT (Devlin et al., 2019) with an input patch size of $32 \times 32$ pixels.

Table 1 summarizes the main concepts of the four architectures side by side.

### *1.3. Random Initialization*

In this scenario, the weights of the models were randomly initialized. This is specific to each architecture and is defined in the corresponding constructor functions of the torchvision package. Thus, the way each model was initialized can be found in the code of its implementation at `https://github.com/pytorch/vision/tree/main/torchvision/models`.

### *1.4. Learning rate*

The starting learning rate for each model/training combination was determined using the learning rate range test Smith (2017). Before the actual model training, tests are performed, where the learning rate is increased linearly between two boundaries. Figure 1 shows the loss curve for the VGG-19 BN network with the learning rate in the range of $[0.001 - 0.015]$. Starting with a low initial learning rate, the network starts to converge. As the learning rate is increased, it will get too large leading to the learning of a suboptimal set of weights, and the network diverges. Typically, a good static learning rate can be found half-way on the descending loss curve or at the point with the steepest gradient (minimal gradient) which can serve as a first indicator (cf. dot in Fig. 1).

Table 2 gives an overview of which specific learning rates were used for the different scenarios.

Table 1: Main concepts of the four model architectures used in our study.

| | **VGG** | **ResNet** | **RegNet** | **ViT** |
|---|---|---|---|---|
| Network architecture | Stack of convolutional layers, pooling layers, and fully connected layers | Stack of convolutional layers followed by residual blocks, pooling layers, and fully connected layers | Series of stages, each of which contains a set of identical bottlenecks, pooling layers, and fully connected layers | Transformer encoder and fully connected layer |
| Feature extraction | Convolutional layers | Residual blocks | Bottlenecks | Transformer encoder |
| Long-range dependencies | Learned implicitly | Learned implicitly | Learned implicitly | Learned explicitly |

Table 2: Start learning rates used for pre-training and fine-tuning of the models, that yield the results reported in Tab. 4.

| **Training scenario** | **VGG-19 BN** | **ResNet-152** | **Regnet_y_32gf** | **ViT_l_32** |
|---|---|---|---|---|
| pre-training | | | | |
| PCam | 0.0034 | 0.0075 | 0.0040 | 0.0100 |
| CD | 0.0150 | 0.0300 | 0.0142 | 0.0110 |
| ImageNet + PCam | 0.0034 | 0.0025 | 0.0029 | 0.0010 |
| ImageNet + CD | 0.0013 | 0.0012 | 0.0035 | 0.0010 |
| fine-tuning | | | | |
| Random | 0.0043 | 0.0130 | 0.0040 | 0.0140 |
| ImageNet | 0.0040 | 0.0041 | 0.0066 | 0.0075 |
| PCam | 0.0043 | 0.0043 | 0.0200 | 0.0120 |
| CD | 0.0090 | 0.0100 | 0.0075 | 0.0075 |
| ImageNet + PCam | 0.0019 | 0.0010 | 0.0035 | 0.0010 |
| ImageNet + CD | 0.0020 | 0.0020 | 0.0020 | 0.0020 |

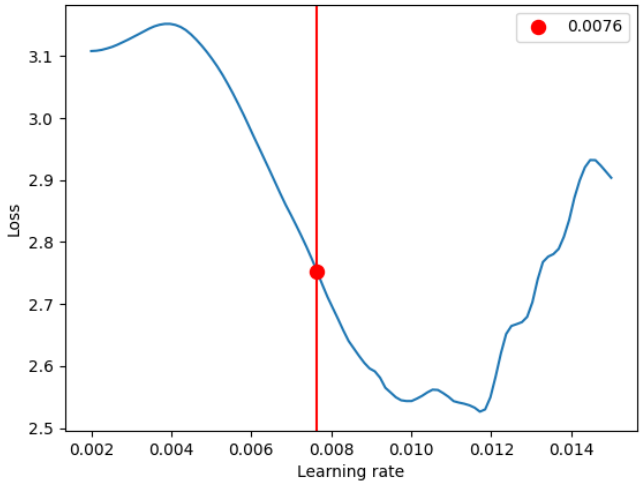


Fig. 1: Validation loss for the VGG-19 BN network for different learning rates. A reasonable learning rate for the model lies half-way on the descending loss curve, or the point with the steepest gradient.

### *1.5. Gradient-weighted Class Activation Mapping*

For the problem at hand, we decided to use an attribution method that is often used in computer vision. The basic idea is to visualize the relevant regions in the input given a models' output. Attribution techniques can be divided into three classes, gradient-based, structure-based, and surrogate and sampling-based (Schlegel and Keim, 2021).

Gradient-based methods provide explanations by performing a single forward and backward pass in the network to compute class activation maps (CAMs). These CAMs provide explanations that highlight the regions in the input data that have the most influence on a model's output. Thus, a CAM can highlight regions in the input image that are maximally representative of its class.

Gradient-weighted Class Activation Mapping (Grad-CAM) (Selvaraju et al., 2017) uses the gradients of any target class flowing into the final convolutional layer of a CNN to produce a coarse localization map that highlights the important regions in the input for class prediction. Unlike previous approaches, Grad-CAM is applicable to a wide variety of CNN model families without architectural changes or re-training. The reason for using the last convolution layer is that it is expected to provide the best compromise between high-level semantics and detailed spatial information. The neurons in this layer look for semantic, class-specific information in the input, while the spatial information is lost later in the fully connected classification layer(s).

The algorithm works like this: Given an input signal and a class of interest, we forward propagate the signal through the CNN part of the model and then, through task-specific com-

putations, obtain a raw score for the category. The gradients are set to zero for all classes except the desired class, which is set to 1. This signal is then backpropagated to the rectified convolutional feature maps of interest, which we combine to compute the coarse Grad-CAM localization, which represents where the model must look to make the particular decision. Finally, we pointwise multiply the heatmap with guided backpropagation to obtain guided Grad-CAM visualizations that are high-resolution and concept-specific (Selvaraju et al., 2020).

Chattopadhay et al. (2018) proposed Grad-CAM++ to provide better visual explanations (when compared to Grad-CAM), both in terms of better object localization and explaining occurrence of multiple objects of a class in a single image. Grad-CAM++ uses a weighted combination of the positive partial derivatives of the last convolutional layer feature maps with respect to a specific class score as weights to generate a visual explanation for the class label under consideration.

We looked at the heatmaps generated by Grad-CAM and Grad-CAM++ for the BAS and FGC classes, which are discussed further in the Results section. In our case, the heatmaps of both algorithms are very close to each other. Furthermore, the Grad-CAM++ heatmaps did not provide more meaningful heatmaps. Therefore, we chose the simpler and faster Grad-CAM algorithm. Figures 2 and 3 show some examples side by side.

One reason that Grad-CAM works well may be the fact, that there is usually only a single object/cell in the images. Also, Grad-CAM can sometimes be more informative if one is interested in identifying the most important features for a particular prediction, since it does not weight the gradients based on their proximity to the center of the object.

## References


Boughorbel, S., Jarray, F., El-Anbari, M., 2017. Optimal classifier for imbalanced data using matthews correlation coefficient metric. PLOS ONE 12, 1–17. doi:10.1371/journal.pone.0177678.

Chattopadhay, A., Sarkar, A., Howlader, P., Balasubramanian, V.N., 2018. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks, in: 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 839–847. doi:10.1109/WACV.2018.00097.

Devlin, J., Chang, M.W., Lee, K., Toutanova, K., 2019. BERT: Pre-training of deep bidirectional transformers for languageunderstanding, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), Association for Computational Linguistics, Minneapolis, Minnesota. pp. 4171–4186. doi:10.18653/v1/N19-1423.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N., 2021. An image is worth 16x16 words: Transformers for image recognition at scale, in: International Conference on Learning Representations.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778.

LeCun, Y., 2012. Learning invariant feature hierarchies, in: Fusiello, A., Murino, V., Cucchiara, R. (Eds.), Computer Vision – ECCV 2012. Workshops and Demonstrations, Springer Berlin Heidelberg, Berlin, Heidelberg. pp. 496–505.

Matthews, B., 1975. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. Biochimica et Biophysica Acta (BBA) - Protein Structure 405, 442–451. doi:https://doi.org/10.1016/0005-2795(75)90109-9.

Radosavovic, I., Kosaraju, R.P., Girshick, R., He, K., Dollár, P., 2020. Designing network design spaces, in: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 10425–10433.

Schlegel, U., Keim, D.A., 2021. Time series model attribution visualizations as explanations, in: 2021 IEEE Workshop on TRust and EXpertise in Visual Analytics (TREX), pp. 27–31. doi:10.1109/TREX53765.2021.00010.

Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D., 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization, in: 2017 IEEE International Conference on Computer Vision (ICCV), pp. 618–626.

Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D., 2020. Grad-cam: Visual explanations from deep networks via gradient-based localization. International Journal of Computer Vision 128, 336–359. doi:10.1007/s11263-019-01228-7.

Sick, L., 2021. Regnet: The most flexible network architecture for computer vision. URL: https://towardsdatascience.com/regnet-the-most-flexible-network-architecture-\-for-computer-vision-2fd757f9c5cd.

Simonyan, K., Zisserman, A., 2015. Very deep convolutional networks for large-scale image recognition, in: International Conference on Learning Representations.

Smith, L.N., 2017. Cyclical learning rates for training neural networks, in: 2017 IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 464–472.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I., 2017. Attention is all you need, in: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (Eds.), Advances in Neural Information Processing Systems, Curran Associates, Inc.

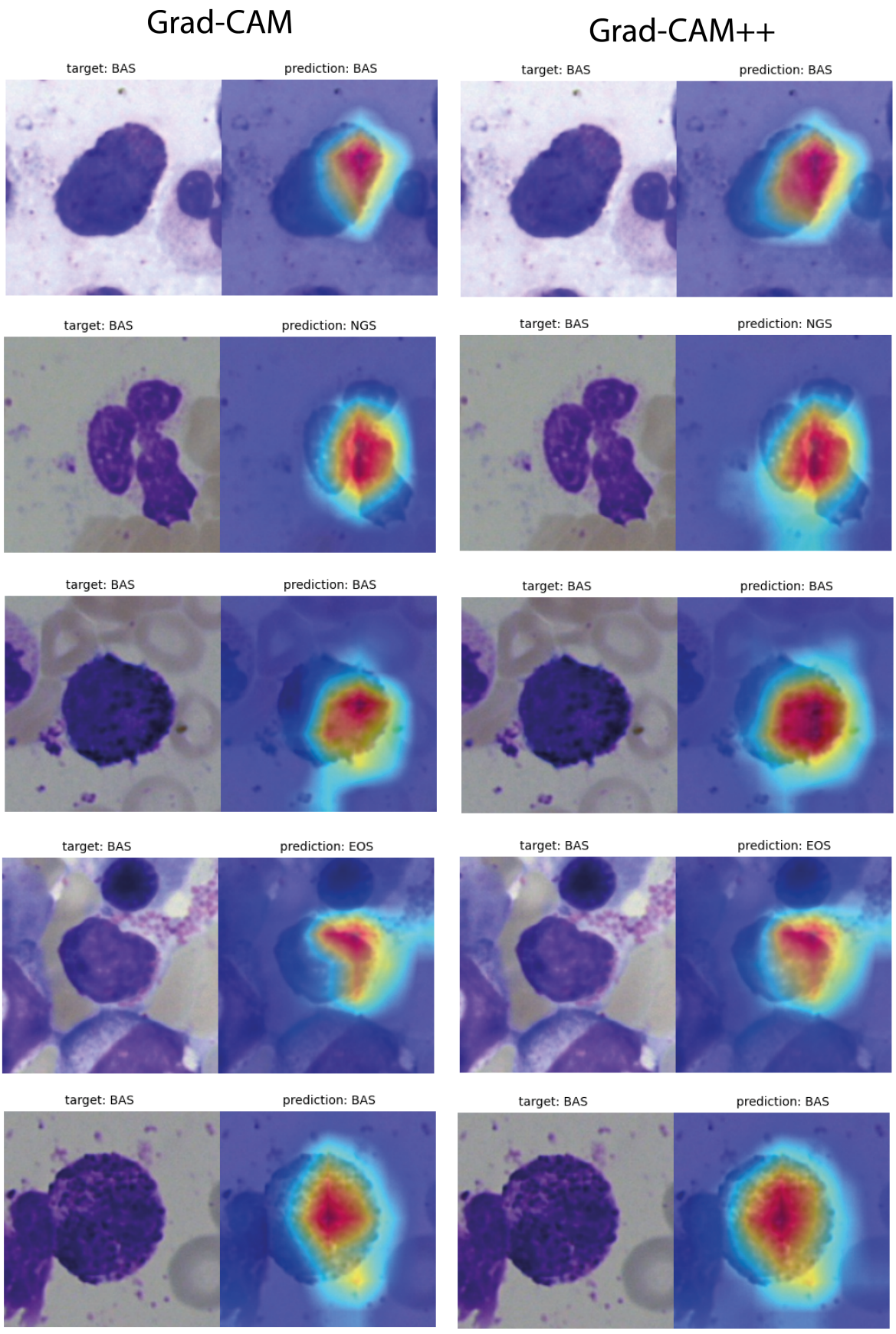


Fig. 2: Samples of Grad-CAM (left) and Grad-CAM++ (right) activation maps generated from the Basophils using the Regnet_y_32gf pre-trained on ImageNet + CD. Regions showing high activation (in red) provide a strong contribution to the classification result.

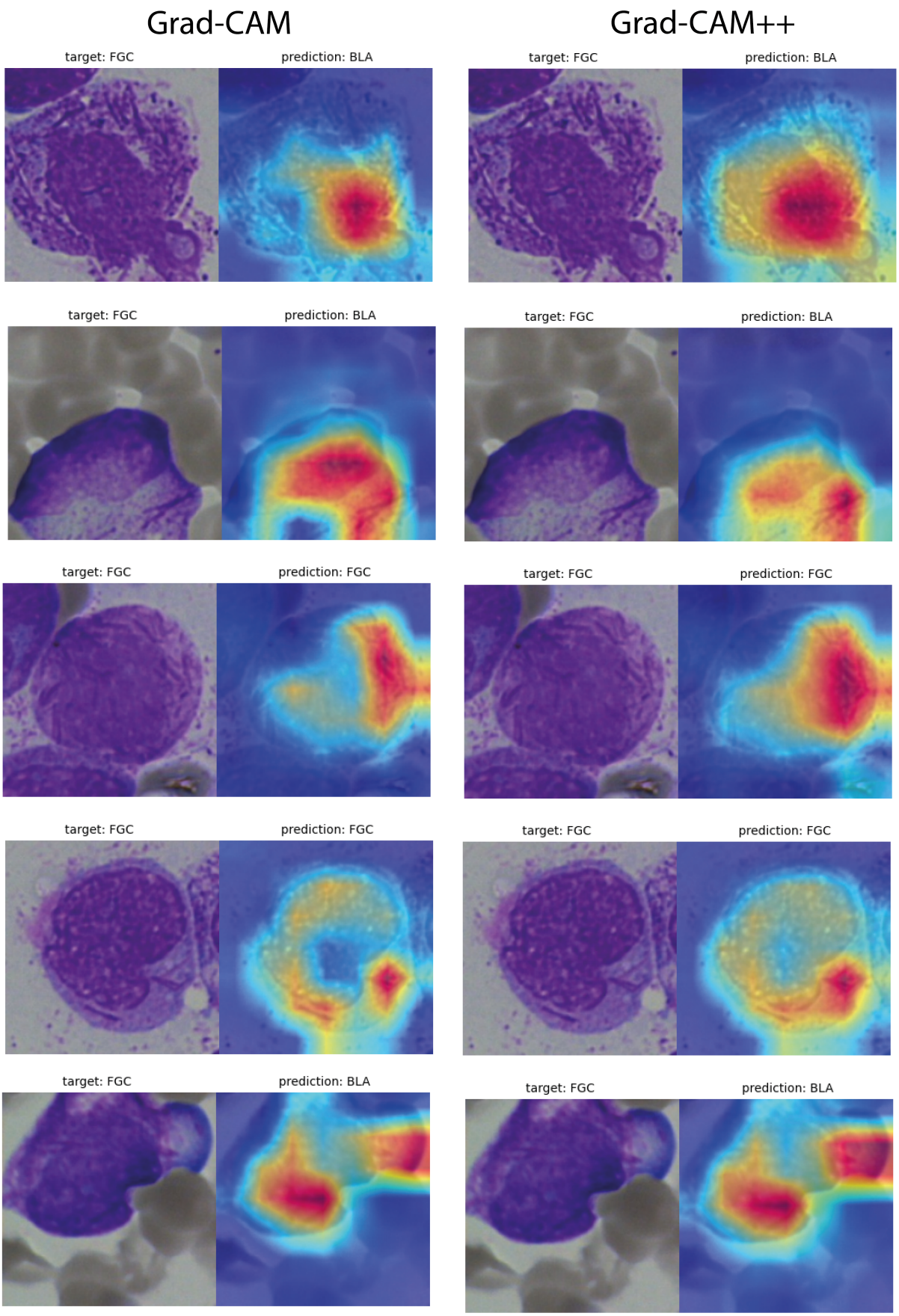


Fig. 3: Samples of Grad-CAM (left) and Grad-CAM++ (right) activation maps generated from the Faggot cells using the Regnet_y_32gf pre-trained on ImageNet + CD. Regions showing high activation (in red) provide a strong contribution to the classification result.