Deep-learning (BMEVITMMA19)

Project work documentation

Hermán Judit (L7D38R)
Kovács Kíra Diána (CXL05B)

1 Introduction

Machine learning, particularly deep learning, has become a cornerstone in advancing medical imaging and diagnostics. Convolutional neural networks (CNNs) and other deep learning models excel in analyzing medical images like MRI, CT, and X-ray scans by autonomously learning patterns without manual feature extraction. These techniques are widely applied to classify diseases, detect abnormalities, and segment anatomical structures across disciplines such as radiology, pathology, and cardiology. Despite their success, challenges like the need for large annotated datasets, computational demands, and model interpretability remain significant. [1] [2]

Additionally, deep learning aids in multi-modal data integration (e.g., combining imaging and genetic data), personalized medicine, and even generating synthetic medical images to augment datasets. Techniques like transfer learning are also being used to address data scarcity by adapting models trained on large general datasets for specific medical applications. [2] [3]

In the framework of this project we classified chest x-ray images into normal, viral and bacterial pneumonia classes. We compared several different models: convolutional and transformer models, both from scratch and using transfer-learning.

1.1 Data

The data we used is a Kaggle dataset [4] of Chest X-ray Images and originally contains 2 classes: normal lung and pneumonia. But the filenames of the pneumonia images contained wether the pneumonia was caused by virus or bacteria, so we did a 3-class classification instead of a binary classification. (There was a Kaggle competition on the data for binary classification.) The dataset contains more than 5000 grayscale images in various sizes and their labels divided into train and test sets. 1

Originally, the bacteria class contained twice as many images as the other two, but for the most efficient training we needed nearly equal class sizes. The most proper way of handling inbalanced data would be to oversample the smaller classes with data augmentation, but because of the limited computational capacity, we downsampled the bacteria class. 2

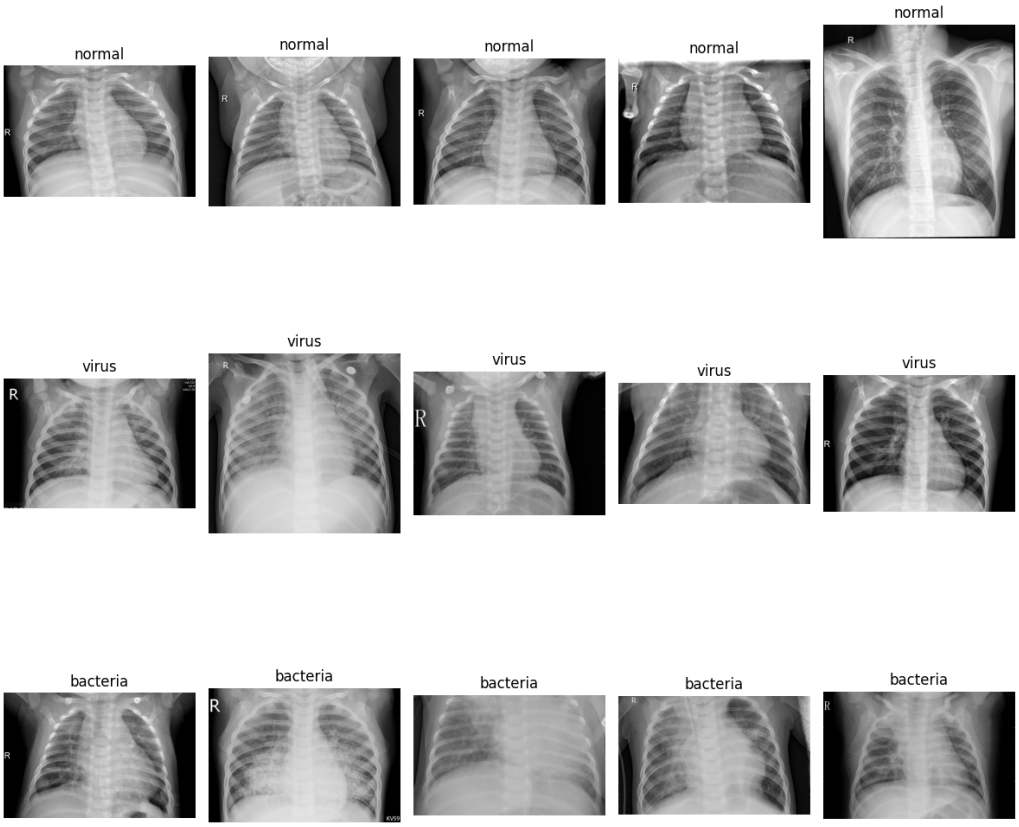

Figure 1: Sample images from the 3 classes.

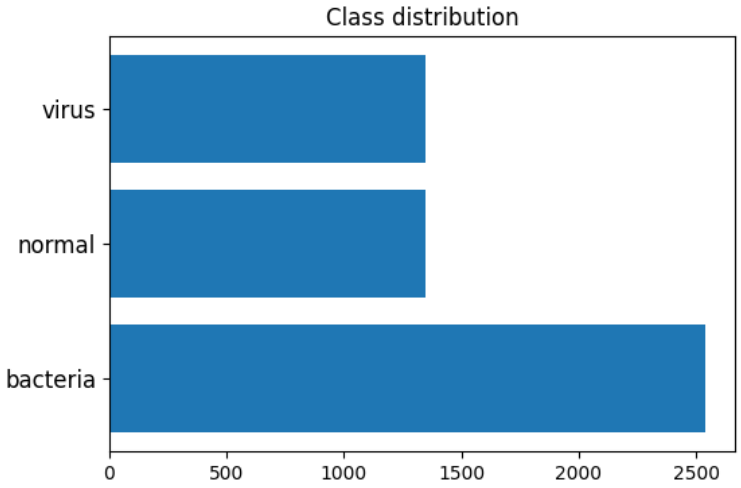

Figure 2: Inbalanced class sizes.

2 Methods of training

Our aim was to try out different deep learning models with several hyperparameters to learn how to use them, compare them, and find out which one is the most appropriate for this problem. We trained our models on the server of Kaggle using GPU, but the GPU time was limited, so we could not use hyperparameter optimization. We did the training in Pytorch and used CrossEntropyLoss.

2.1 Hyperparameters, data preparation

We used a learning rate of 10^{-3}, a batch size of 128 and Adam optimizer. Because we mostly used pre-trained models which were trained on ImageNet, we used $224x224$ image size, tripled the channel size because the ImageNet dataset contains colorful images and normalized them. For most of the training we used CosineAnnealingLr as learning rate scheduler, because it made the models better by our experience. [5] 3

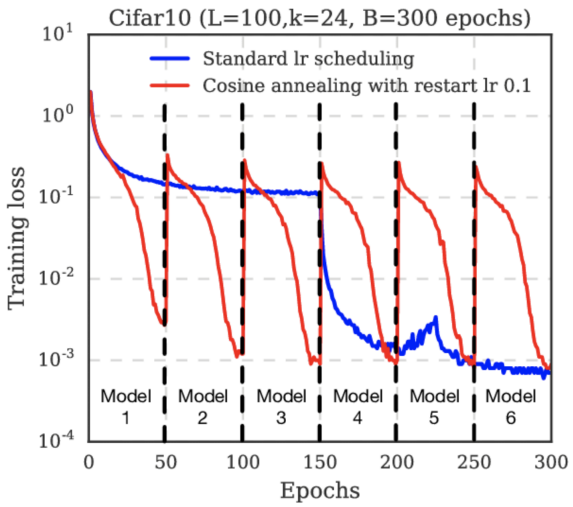

Figure 3: Cosine Annealing Scheduler.
Source: https://paperswithcode.com/method/cosine-annealing

We trained the models through 20-25 epochs and used early stopping with patience of 2 epochs and saved only the states of the models. For some models we tried AdamW optimizer with a few different values of weight decay, but it did not result an improvement in the models' performance.

2.2 Models

As a baseline we used a simple convolutional network which contained 2 2-dimension convolution layers, a 2x2 pooling layer and 2 fully connected layers, with relu activation. This is a very simple, yet quite effective model for image classification tasks. The training stopped after 8 epochs because of early stopping, the training accuracy was 0.91 and the validation accuracy 0.77 in this last epoch, so we can say, that the model overfitted for the training data. 4

After the baseline, we experimented with an other convolutional network that we trained from scratch. It had 5 convolutional layers, a pooling layer, 3 fully connected layers, and a dropout layer, all with relu activations. The training stopped after 12 epochs, the training accuracy was 0.85 and the validation accuracy was 0.79 there. So this model is still overfitted, just not as much as the baseline model, which is surprising, because it is a much more complex network. 5

We tried out different pre-trained convolutional models: ResNet50, EfficientNet-B0 and MobileNet-V2. (And we also used VGG16, but it it was very slow and performed not so well with this number of epochs, so we continued to work with the other 3 pre-trained convolutional networks.) For these models we used CosineAnnealingLr scheduler. First, we freezed the weights of the models and traind only the last layers, then fine tuned them through a few (3-5) epochs.

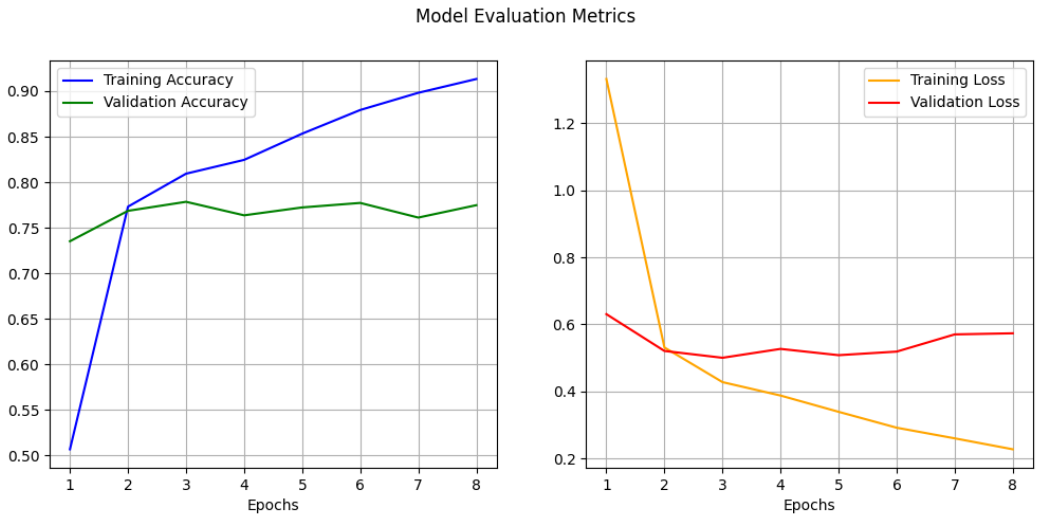

Figure 4: Baseline model accuracy and loss.

Figure 5: First CNN model accuracy and loss.

ResNet reached the validation accuracy of 0.78 with frozen weights and during fine tuning it became higher than 0.8. 10

Then we trained the same model using weight decay to find out if it gets better, but it only got worse.

The efficientNet-B0 turned with Adam optimizer without weigt decay reached also the 0.78 validation accuracy and got better with fine tuning, but it overfitted. 11

So we trained it again using weight decay of $5 * e^{-}4$ and AdamW optimizer, and this was the model with the best test accuracy.(Despite the fact that it also overfitted.) 8

MobileNet-V2 had slightly lower accuracy than the last two models, but even it exceeded 0.8 in the validation set. 9

Finally, we used vision transformer models both pre-trained and trained from scratch.

For the made from scratch model, we defined 4 classes. The first class was for patch embeddings, which divides pictures into non-overlapping patches. The next class is for Multi-Head

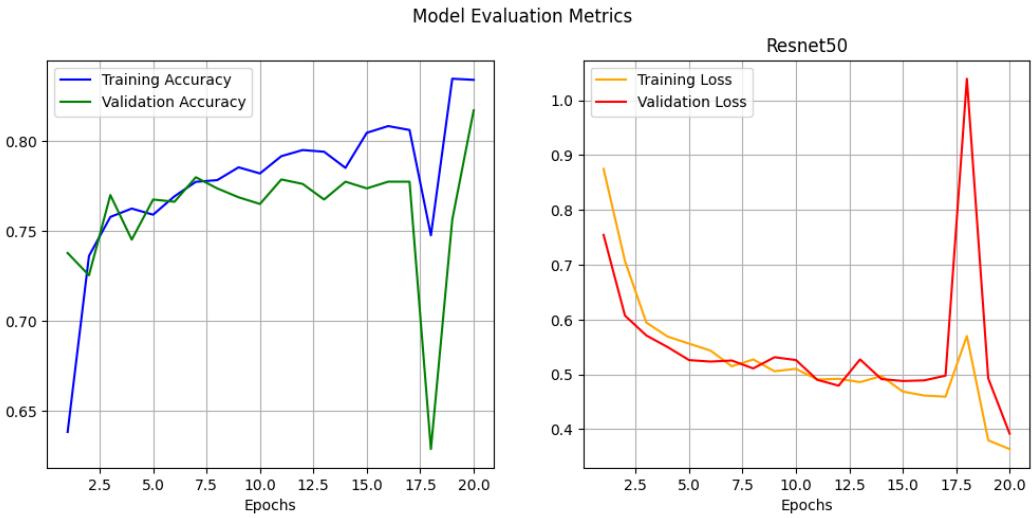

Figure 6: ResNet50 model accuracy and loss.

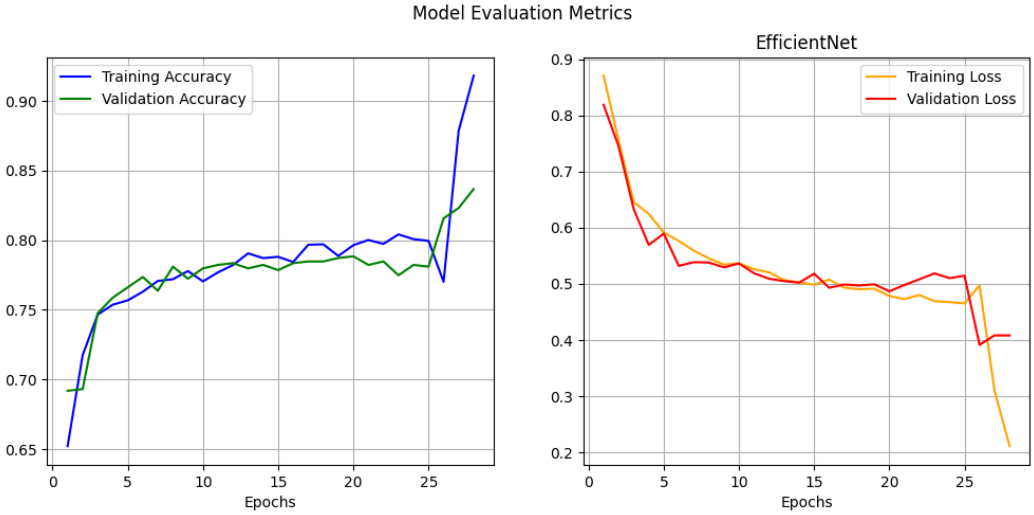

Figure 7: EfficientNet-B0 model accuracy and loss.

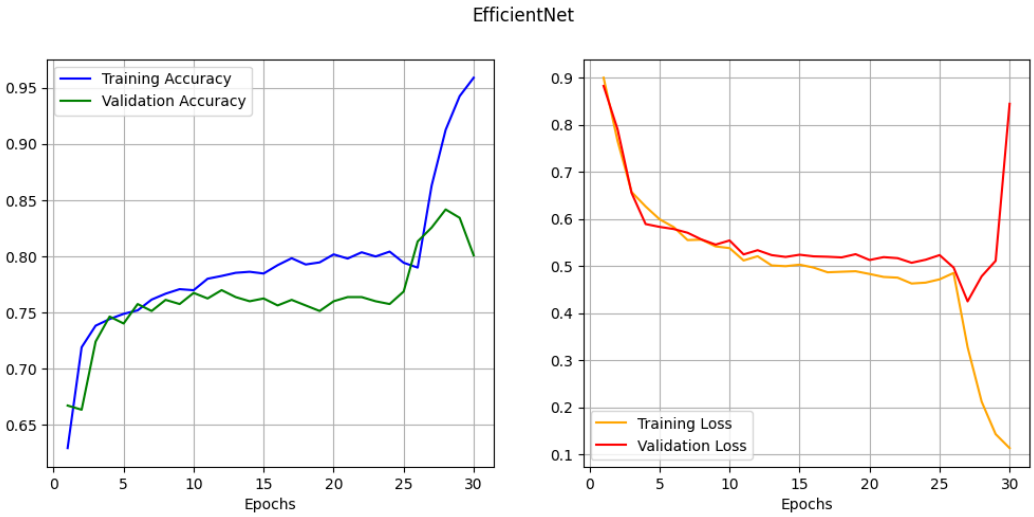

Figure 8: EfficientNet-B0 model with weight decay accuracy and loss.

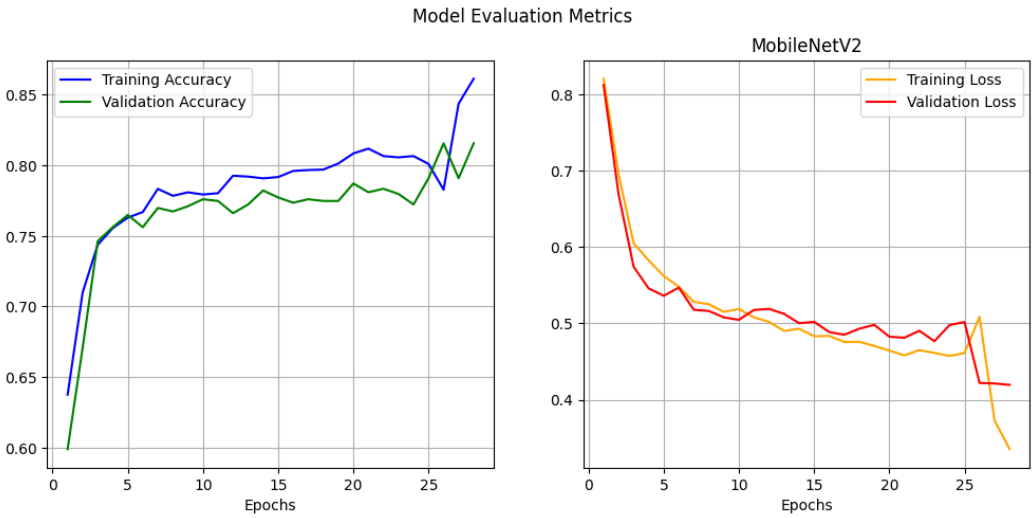

Figure 9: MobileNet-V2 model accuracy and loss.

Self-Attention (MHSA), which extracts relationships between patches. Then we defined a class for a Transformer Block, this is the core of the transformer model. Each block contains an MHSA part, a Feed-Forward Network, which is a two-layer perceptron with GELU activation, and residual connections and LayerNorm to stabilize the model. The final class is the Vision Transformer itself, which is built from the 3 classes mentioned before. The training of this transformer model stopped after 15 epochs, the training accuracy was 0.6 and the validation accuracy was 0.63 in that last epoch. 10

Figure 10: From-scratch transformer model accuracy and loss.

After this transformer model, we wanted to try a pre-trained one as well, we chose vit_base_patch16_224 [9] for it. It is trained on ImageNet as well. The training of this model ran through all 20 epochs, the training accuracy was 0.76 and the validation accuracy was 0.71 in the last epoch, so it is better, than our from-scratch transformer model, which is not so surprising. 11

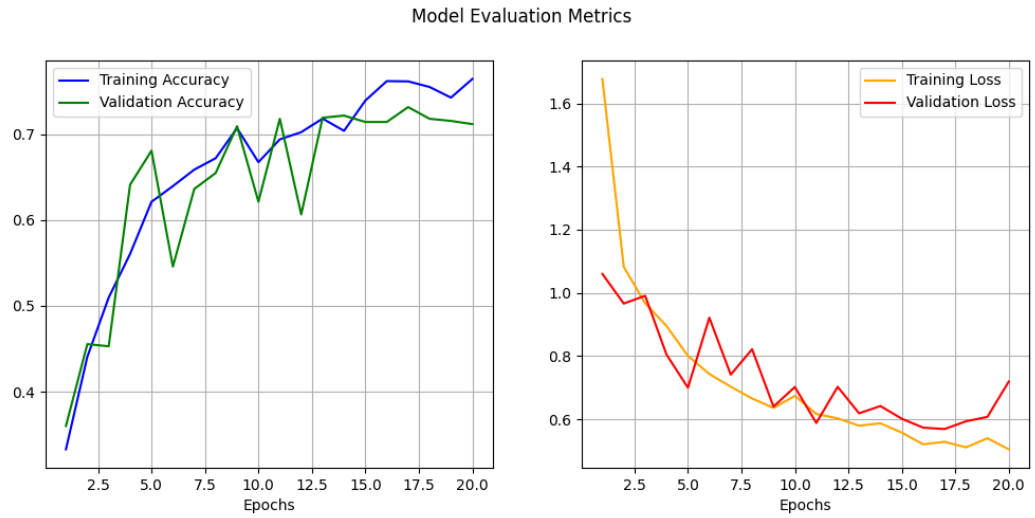

Figure 11: Pre-trained transformer model accuracy and loss.

3 Evaluation

We evaluated our models on the test set and got the following results: the best performing model was Efficientnet with weight decay of $5 * 10^{-4}$ with 88% test accuracy (which is not bad for 3 classes). The second best model was Resnet without weight decay with almost 86% test accuracy. The EfficientNet model which was trained without weight decay has 82%, and the MobileNet model has 79% test accuracy.

We visualized the confusion matrices of the models. Speaking of the best model, we cannot say that it consistently confuses two classes, so the two pneumonia classes are not more similar than the normal class for it. We can say, that what it consideres as normal case, it belongs to normal class, because there are only two exceptions in the test set which were misclassified as normal.

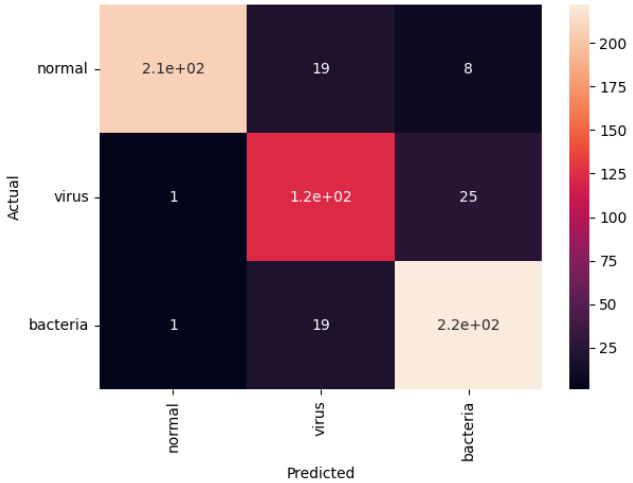

Figure 12: Confusion matrix of EfficientNet-B0 with weight decay.

For the other models (the complex CNN, and the 2 transformer models), we didn't get a normal test accuracy result, because something was wrong with the values. As we could see in the previous section, the validation accuracy was well above the random guessing, but for some reason on the

test data we got around 0.33 accuracies. We tried to figure out, what is the problem, since the model worked well for the validation data, which it didn't see previously as well, and we got good results for the pre-trained CNN models, so the test accuracy calculation should be good too. But until the deadline we couldn't figure out the problem in the code.

4 Conclusions

This project explored deep learning models for classifying chest X-ray images into normal, viral pneumonia, and bacterial pneumonia. Pre-trained models like EfficientNet-B0 with weight decay performed best, achieving 88% test accuracy, highlighting the value of transfer learning. Despite successes, challenges like overfitting and imbalanced data remain, limiting generalizability. Future work could involve integrating additional regularization techniques, exploring semi-supervised learning, or leveraging synthetic data to enhance model robustness.

Usage of LLMs

We used the help of ChatGPT to write the introduction, conclusion and search related scientific papers.

References

[1] Aimina Ali Eli and Abida Ali. *Deep Learning Applications in Medical Image Analysis: Advancements, Challenges, and Future Directions.* arXiv, 2024.

[2] Li, Mengfang and Jiang, Yuanyuan and Zhang, Yanzhou and Zhu, Haisheng. *Medical image analysis using deep learning algorithms.* Frontiers in Public Health, 2023.

[3] Ker, Justin and Wang, Lipo and Rao, Jai and Lim, Tchoyoson. *Deep Learning Applications in Medical Image Analysis.* IEEE Access, 2018.

[4] https://www.kaggle.com/datasets/tolgadincer/labeled-chest-xray-images/data

[5] https://pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.CosineAnnealingLR.html

[6] https://pytorch.org/vision/0.18/models/generated/torchvision.models.resnet50.html

[7] https://pytorch.org/vision/main/models/generated/torchvision.models.efficientnet_b0.html

[8] https://pytorch.org/hub/pytorch_vision_mobilenet_v2/

[9] https://huggingface.co/google/vit-base-patch16-224