Applying Neural Networks to Tumor Malignancy Classification
Final report

Murad Aghazada, Erkhes Nyamsaikhan, Kamran Gasimov, Alexander Sharipov
(20200798) (20190786) (20190772) (20190793)

1. Motivation

Cancer is the second leading cause of death globally, taking the lives of over 8 million people every year. Tumor malignancy testing is a major part of cancer diagnosis since the treatment plans for benign and malignant tumors are vastly different. Therefore, high precision and high recall of tumor diagnosis are crucial for identifying proper treatment.In this project, apply neural networks with convolutional and transformer architectures to learn tumor representations and use them for classifying the tumor as either benign or malignant.

2. Methods

In this project, we utilized and compared convolutional and transformer deep learning architectures for lymph node cancer detection[(1)]. To train the neural networks, we used PatchCamelyon (PCam) dataset[(2)]. The original PCam consists of 327.680 color images (96x96px) extracted from histopathologic scans of lymph node sections. Each image is annotated with a binary label indicating the presence of metastatic tissue. Positive label indicates that the center 32x32px region contains malignant tissue whereas the negative label means that the central 32x32 patch doesn't contain a tumor. Due to the large size ofthe original dataset, we decided to use a small fraction of the data points. In the initial round of training, we used 6763 images sampled from PCam, with 90% of the dataset reserved for training, 5% for validation and 5% for testing. In the subsequent rounds of training, we increased the dataset size to 10000 and used 80% for training, 10% for validation, and 5% for testing.
For the training data, we utilized data preprocessing to reduce computational complexity. We re-scaled the numeric values of pixels in all datasets to lie between 0 and 1. Pixel value rescaling was done to reduce computation complexity for both forward and back propagation and reduce the risk of overflow. As for data augmentation, we applied random rotation between 0º-45º , horizontal, vertical flips, shifted image along X and Y axis by 30%, and finally, used zoom-in and zoom-out methods. The main reason to do these kind of data augmentations is to make our trainingdataset as rich as possible, in other words, to make the dataset more representative.

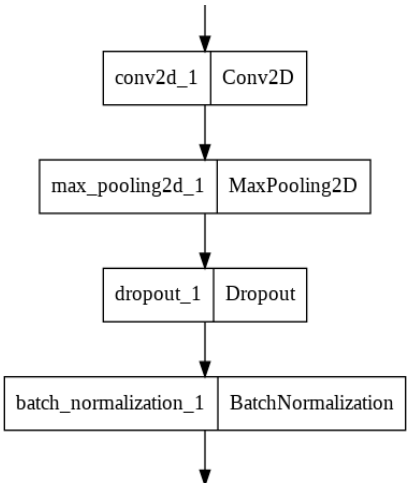

Figure 1. Custom CNN block

We then proceeded to train multiple neural networks. First, we trained a custom network with multiple convolutional layers as the baseline. This model doesn't use any pre-trained network and has 2.1 million trainable parameters. The architecture of the baseline network consists of 4 repeating blocks depicted in Figure 1. First comes the convolutional layer which applies multiple convolutional filters to create a feature map that reveals whether the learned features are present in the input. Then comes a maximum pooling layer that down samples the feature map. This has an effect of making the feature map less sensitive to changes in position of the detected feature and introduces local transition invariance. Next comes the dropout layer which is used to perform model averaging and regularization by randomly dropping its constituent units and their connections. Finally, there is the batch normalization layer which performs re- centering and re-scaling of its inputs. This makes the subsequent computations faster andmore numerically stable by reducing the chance of overflow. The output of the 4 CNN blocks is flattened to a single vector and then interpreted by themultilayered perceptron with 4 dense layers depicted in Figure 3. MLP accepts the flattened feature vector learned by the preceding CNN blocks and then makes the decision on whether the input image contains malignant tumor in the central patch.

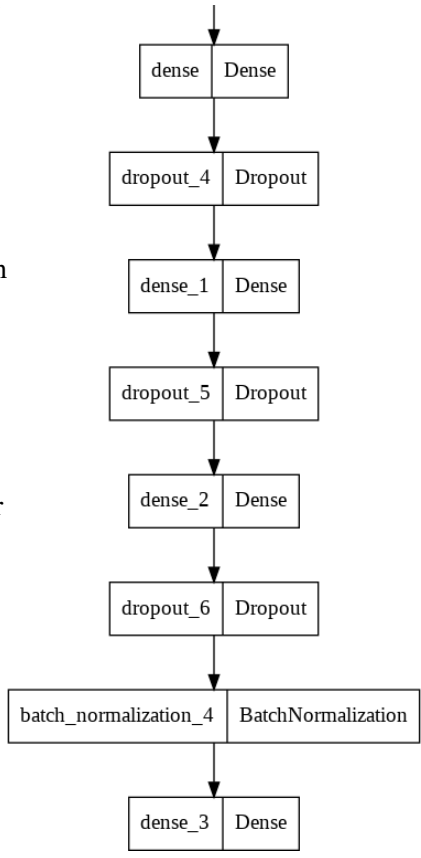

Figure 2. MLP block

Apart from the custom CNN baseline, we utilized transfer learning and constructed 3 more networks which utilize VGG16 (14.8 million parameters), VGG19 (20.0 million parameters), InceptionV3 (21.8 million parameters) and ResNet-50 (23.5 million parameters) as base models initialized with weights from the ImageNet database[(8)]. All layers of the pre-trained models were frozen. All 3 networks based on the pre-trained models take the output of the pre-trained models (excluding their fully connected layers), flatten it to a vector and feed it to the MLP in Figure 2. We chose to use the same MLP upper prediction layer across all tested architectures to make them more comparable. This way, we can directly relate the differences in the overall accuracy to the network's ability to extract useful features from images.

In addition to the custom and pre-trained CNN models, we decided to utilize the transformer encoder-decoder architecture for malignancy classification. Transformer architecture[10] introduced in 2017 revolutionized the NLP field and significantly improved the state-of-the-art NLP performance thanks to the utilized multi-head attention mechanism which relates different positions of a single sequence to compute its representation. Attention takes query ($Q \in R^{d_k}$), key ($K \in R^{d_k}$), and value ($V \in R^{d_v}$) vectors and computes a weighted sum of values. Weight assigned to each value is computed by the compatibility function of query with corresponding key:

$$Attention(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right) \qquad \textbf{(1) Scaled dot-product attention function}$$

The scaling factor $\sqrt{d_k}$ is introduced to increase numeric stability and avoid vanishing gradients at high dimensions of Q, or K
In multi-head attention with ***h*** heads, scaled dot product attention is applied in parallel to ***h*** different learned projections of query, key, and value vectors. ***h*** attention outputs for each of these projections are then concatenated and projected back the original dimension:

$$MultiHead(Q,K,V) = Concat(head_1, \dots, head_h)W^O \qquad \textbf{(2) Multi-head attention function}$$
$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \qquad \textbf{(3) Equation of each head in multi-head attention}$$

Here weight parameters W_i cast the *Q, K, V* vectors to a reduced dimension, producing ***h*** different projections. Meanwhile, W^O introduces non-linearity and additional learnable parameters to produce an optimized overall attention value:

$$W_i^Q \in R^{d_{model}*(\frac{d_{model}}{h})}, W_i^K \in R^{d_{model}*(\frac{d_{model}}{h})}, W_i^V \in R^{d_{model}*(\frac{d_{model}}{h})}, \; W^O \in R^{d_{model}*d_{model})}$$

Multi-head attention allows the transformer to attend to multiple regions of the input sequence at once and thus compute a more complete learned representation of the input. It is the basis of the transformer. On a higher level, transformer consists of 2 key blocks: transformer encoder, and transformer decoder. Transformer encoder produces a representation of the input sequence by applying multi-head self-attention. Meanwhile, the decoder generates the result by first applying the multi-head self-attention to the output of the previous decoder layer, and then applying encoder-decoder attention which uses encoder final state output as keys and values, and the output of the previous decoder as queries. Multiple layers of encoder and decoder may be stacked together to learn deeper patterns from the input data. However, despite its merits, transformers are typically applied to sequential data, such as text, and not images. To make images compatible with the transformer architecture, we patched the 32*32 cropped image into 196 patches, 5*5 pixels each. This procedure is demonstrated below:

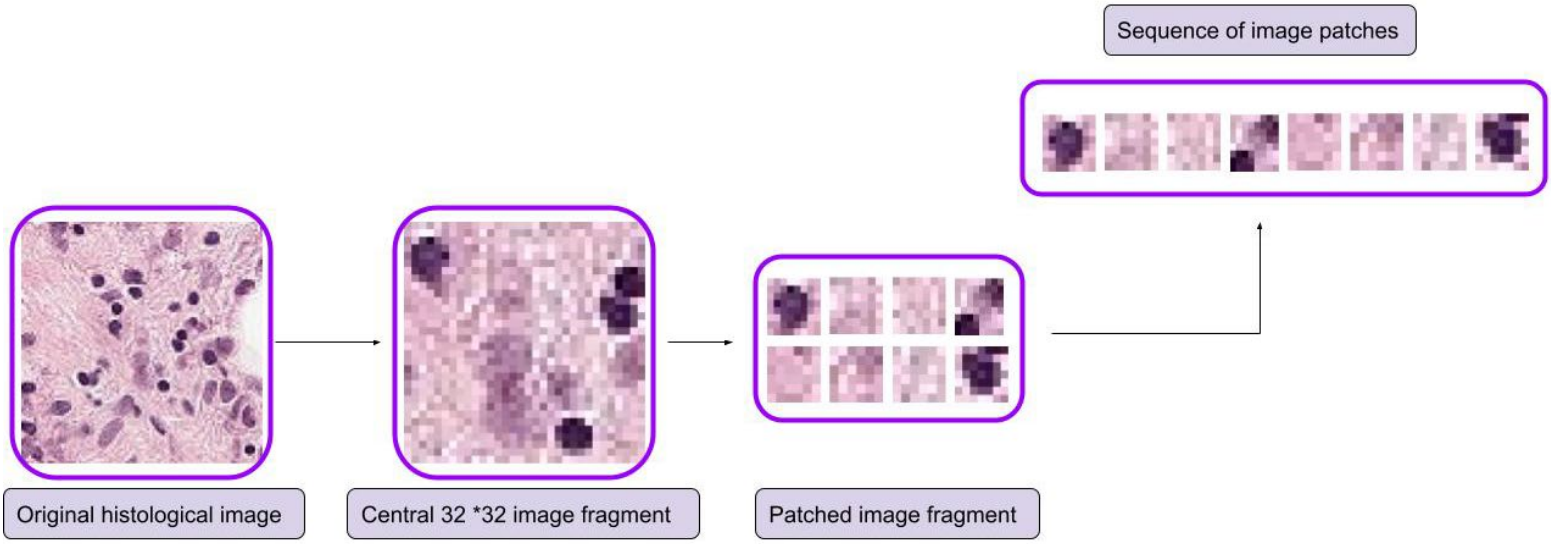

*Figure 4. Image pre-processing for transformer encoder-decoder architecture. The image is first center-cropped to produce 32*32 fragment. The fragment is then patched into 5 * 5 pixel segments and the segments are reshaped into a sequence. Here we use an image from the utilized PCam dataset*

We then reshaped the patches into a sequential vector with 196 patches and used it as input for a simple 1-layer encoder decoder transformer illustrated below. Our transformer model has just 1 encoder and decoder layer and 15 million trainable parameters.

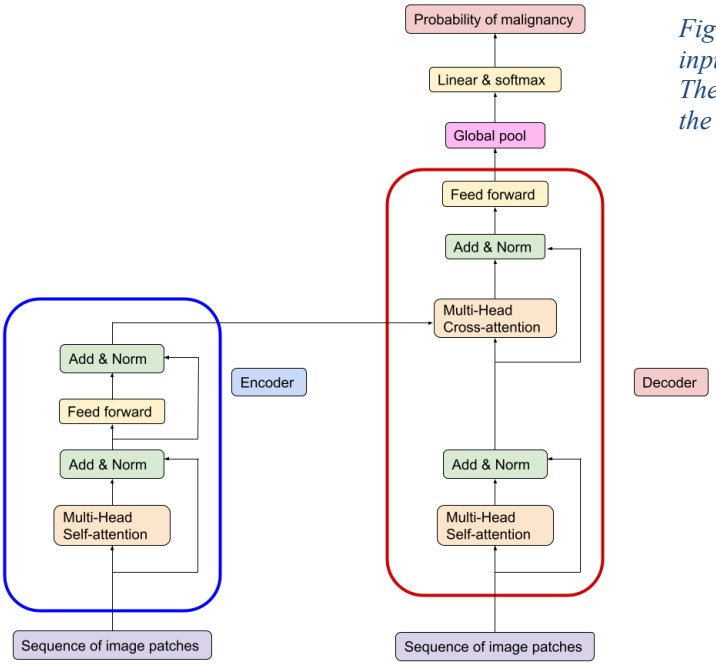

Figure 4. Transformer encoder-decoder for malignancy classification. The input image is patched and reshaped into a vector, then fed into the encoder. The output of the encoder is then fed into the decoder's cross-attention. Finally, the result is pooled and fed into the softmax classifier

The input image is center cropped and then patched and reshaped into a vector, which is subsequently fed into the encoder. The encoder consists of 2 key sub-layers. The first sub-layer computes multi-head self-attention based on the sequence of image patches. The second sub-layer is a fully connected feed-forward neural network with 3 dense layers. The residual connection with subsequent normalization is used around each of the 2 encoder layers. The output of the encoder contains the hidden state for each patch in the sequence. The second major element of the transformer is the decoder layer, which consists of 3 key sub-layers. The first sub-layer computes multi-head self-attention based on the sequence of image patches. The second sub-layer computes multi-head encoder-decoder cross-attention based on the self-attention from the previous layer and the encoder output. The third sub-layer is a feed-forward neural network with 3 dense layers. The residual connection with subsequent normalization is used around each of the 3 layers. Finally, the normalized result is pooled to extract the key features and reduce dimensionality. The result of global average pooling is fed into the softmax classifier which predicts the probability of tumor malignancy.

Finally, we also developed a hybrid architecture which combines the benefits of convolutional neural networks for extracting useful features for the data and the benefits of transformer which can learn powerful representations of the input sequence thanks to the multi-head attention mechanism. To do this, we first center-cropped the image and then fed the output into VGG19. We then extracted the output of the fourth convolutional layer of VGG19 which contained 128 feature maps, each 16*16 pixels. The extracted feature maps were then patched into 512 patches, each 2*2 pixels. The patches were then reshaped into a single vector and then fed into the transformer encoder-decoder as is shown in the diagram below:

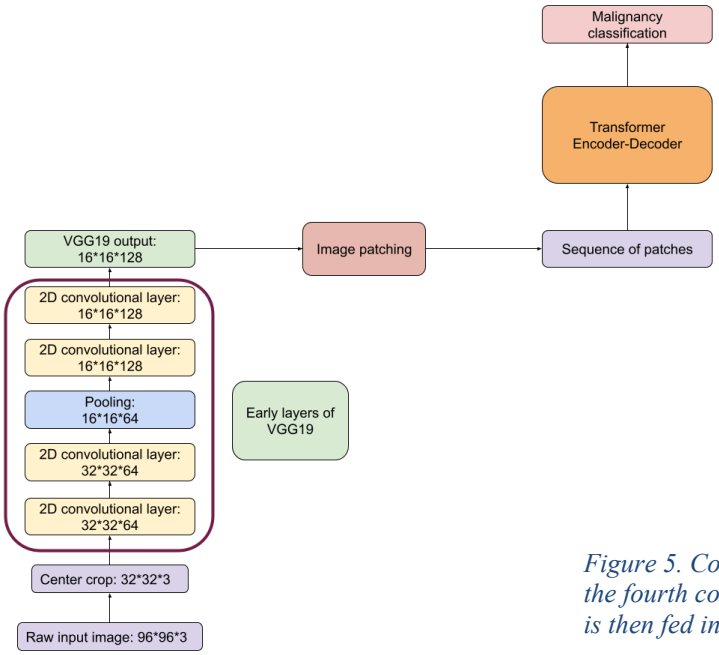

Figure 5. Combined CNN and transformer architectures. The feature map from the fourth convolutional layer is patched and reformed into a sequence, which is then fed into the transformer encoder-decoder

3. Experiments and results

All 6 neural networks were implemented using TensorFlow machine learning framework and were then trained using Google Colab cloud-based computing resource. We trained the baseline CNN model as well as the 4 pre-trained CNN models (VGG16, InceptionV3, ResNet50) for 10 epochs on the training set with pre-processed 6103 images in batches of 64 using Root Mean Squared propagation optimizer with the learning rate 0.001 and the binary cross entropy as the loss function. The validation score is based on the validation set with 343 images and the final accuracy is evaluated on the test set with 317 images. The results of training are presented in Table 1.

Model	Test set accuracy, %	MLP input vector dimension
VGG16	83.5	4608
InceptionV3	76.3	2048
Baseline model	72.6	18432
ResNet50	69.8	18432

Table 1. The results of training

Among all the tested architectures, VGG16 achieves the highest test set accuracy. This could be because we used a relatively small dataset and trained only the top MLP layers of the networks based on the pre-trained models. VGG16 supplied 4608-dimensional vector to the MLP, which worked best since it was neither too small, as in the case of InceptionV3, and neither too large, as in the case of the baseline and ResNet50. The second-best model was InceptionV3, whose 2048-dimensional vector meant fewer learnable parameters and thus lower accuracy. The baseline model scored higher than the pre-trained ResNet50 likely because it was entirely trainable and could better fit the dataset. ResNet50 had the lowest score, likely because its MLP input had a high dimension and because its convolutional layers were not trainable. The training history for all 4 models is presented below. Interestingly, in all networks with pre-trained models, the validation set accuracy eventually surpassed training set accuracy. This is a highly unexpected result. However, it can be explained by excessive data augmentation applied to training data which might have shifted the essential 32x32 central patch, resulting in lower accuracy.
In addition to the models above, we also trained a larger VGG19 with 19 layers. The model was trained for 50 epochs on the training set with pre-processed 8000 images in batches of 64 using the Adam optimizer with the learning rate 1e-4 and the categorical cross entropy as the loss function. The validation score is based on the validation set with 1000 images and the final accuracy is evaluated on the test set with another 1000 images. We decided to evaluate the effect of central cropping on the model's performance and thus we trained VGG19 with and without using central cropping. The results of training are presented in Table 2.

Model	Test set accuracy, %
VGG19 (no cropping)	58.5%
VGG19 (with central cropping)	74.1%

Table 2. Comparison of VGG19 performance with and without central cropping

As evident from the results, central cropping drastically improves the performance of VGG19, likely because it allows the model to focus only on the key central patch which is the only part of the image used for malignancy label.

Next, we trained the transformer encoder-decoder. Further, we wanted to compare the transformer encoder-decoder with the hybrid transformer + CNN architecture. Based on the results from VGG19, we applied central cropping to the original image in both cases to maximize the chance of higher accuracy. Both models were trained for 50 epochs on the training set with pre-processed 8000 images in batches of 64 using the Adam optimizer with the learning rate 1e-4 and the categorical cross entropy as the loss function. The validation score is based on the validation set with 1000 images and the final accuracy is evaluated on the test set with another 1000 images. The results of training are presented in Table 3.

Model	Test set accuracy, %
Transformer	74.8%
CNN + Transformer	77.9%

Table 3. Comparison transformer and hybrid transformer + CNN architecture

The results show a respectable performance of the simple transformer architecture, which only has 15 million parameters but performed better than some of the prior pre-trained CNN models. We can also see that the addition of the first 4 convolutional layers from VGG19 slightly improved the performance, highlighting the power and efficacy of convolutional architectures in extracting features from the image. The results also demonstrate that the CNN can be used in tandem with transformer encoder decoder to reap the benefits of both the feature extraction by CNN and learning strong representation of the data by transformer.

4. Discussion

In this paper, we applied multiple deep learning approaches to the highly clinically relevant problem of tumor diagnosis and malignancy classification. We began by training a custom convolutional neural network with just 4 convolutional blocks and very few (2 million) learnable parameters. Despite the small size of the model, it performed reasonably well, producing a 72.6% test set accuracy.

We then proceeded to follow the transfer learning approach and applied 4 popular pre-trained convolutional neural networks for the malignancy classification task. One of the evaluated architectures, VGG16, produced the best accuracy of all the tested models in this paper at 83.5%. Inspired by this result, we decided to use a larger version of the same model architecture, i.e., VGG19, for the remainder of the project.

First, we tried to improve the performance by performing additional pre-processing step, i.e., central cropping the image and using only the central segment for the malignancy classification. Although this did lead to a significant 15.6% improvement over the baseline VGG19, this still didn't surpass the original 83.5% accuracy of VGG16. However, the relative improvement in the accuracy was still significant. Therefore, we decided to keep the central cropping step for the remaining 2 models.

Finally, we decided to use a novel architecture, transformer encoder-decoder. This architecture revolutionized the NLP field and in addition, transformer models occupy the leading spots in the ImageNet leaderboard for computer vision. So, we decided to replicate a simple transformer with just 1 encoder and decoder layer to try to improve the performance. Since transformers are originally applied to the sequential input, we needed to make the image data compatible with this architecture. To do this, we patched the image into smaller fragments, and then reshaped the resulting matrix into a sequential vector. This way, we applied the transformer architecture to the image classification task. Despite the small size (just 1 encoder and decoder layer, and relatively small number of parameters), the implemented transformer showed respectable 74.8% accuracy which beats some of the convolutional models. Still, this was a suboptimal performance.

So, we decided to modify the transformer by adding 4 early convolutional layers of VGG19 to extract the useful features from the image and then feed the resulting feature maps into the transformer. This significantly improved the transformer accuracy (+3.1%), which highlights the effectiveness of convolution in extracting features from images. This feature of CNNs neatly complements the transformer architecture, whose main advantage is the use of multi head attention for learning a strong representation of the sequential input. Therefore, applying the transformer architecture to an extracted feature map from the CNN allows us to combine the benefits and unique strengths of both architectures, improving the overall test set accuracy.

In conclusion, we applied multiple neural network architectures for tumor malignancy classification. We demonstrated that transformer encoder-decoder can be applied not only to the NLP, but also to the computer vision field. It can be applied to a patched image reshaped into a sequence. In addition, we developed a hybrid CNN-transformer architecture, which managed to outperform the plain transformer. This shows that CNN and transformer architectures are in fact compatible and they can complement each other in a meaningful way to combine benefits of both architectures and produce better results.

5. Contributions

Alexander Sharipov:
Developed and trained the custom baseline CNN from scratch. Applied pre-trained VGG16, ResNet50, and InceptionV3 to malignancy classification. Came up with the idea to apply transformer to images. Implemented the simple transformer from scratch. Came up with the idea to make a hybrid architecture from a CNN and a transformer. Implemented the hybrid architecture. Trained the plain transformer and hybrid architectures. Made all the illustrations for the report. Wrote the motivation part, experiments and results, discussion part. Contributed to references and methods part.
Murad Aghazada:
Performed data pre-processing, including the central cropping and augmentation. Extracted 10000 images from the original dataset and organized the dataset into training, validation, and testing subsets. Organized the google drive for storage and use of the dataset. Applied pre-trained VGG19 to malignancy classification. Contributed to "preliminary experiments" section of the report.
Erkhes Nyamsaikhan:
Researched the background and contributed "motivation" section of the report. Worked on extracting images from the dataset. Applied VGG 16 to malignancy classification. Contributed to the final report.
Kamran Gasimov:
Researched literature background, extracted 10000 images from the original dataset and organized the dataset, performed data pre-processing, including the central cropping.

6. References

1. Reshma, V. K, Arya, N., Ahmad S. S., Wattar, I., Mekala, S., Joshi, S., & Krah, D. (2022). Detection of Breast Cancer Using Histopathological Image Classification Dataset with Deep Learning Techniques. BioMed Research International, 2022, 2314-6133
2. PCam dataset: https://github.com/basveeling/pcam
3. Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. Computer Science > Computer Vision and pattern recognition. https://arxiv.org/pdf/1409.1556.pdf
4. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A. (2014). Going Deeper with Convolutions. Computer Science > Computer Vision and Pattern Recognition. https://arxiv.org/pdf/1409.4842.pdf
5. Jonsson, T., & Tapper, I. (2020). Evaluation of two CNN models, VGGNet-16 & VGGNet-19, for classification of Alzheimer's disease in brain MRI scans. KTH ROYAL INSTITUTE OF TECHNOLOGY. https://www.diva-portal.org/smash/get/diva2:1464104/FULLTEXT01.pdf
6. He, K., Zhang, X., Ren, S., Sun, J. (2015). Deep Residual Learning for Image Recognition. Computer Science > Computer Vision and Pattern Recognition. https://arxiv.org/abs/1512.03385
7. Benedetti, P., Perri, D., Simonetti, M., Gervasi, O., Reali G., & Femminella, M. (2021). Skin Cancer Classification using Inception Network and Transfer Learning. https://arxiv.org/pdf/2111.02402.pdf
8. ImageNet database: https://image-net.org/about.php
9. Kaggle cancer classification competition: https://www.kaggle.com/competitions/histopathologic-cancer-detection/overview
10. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention Is All You Need. Computer Science > Computation and Language. https://arxiv.org/abs/1706.03762