Skin Cancer Detection with 3D-TBP
Rosszindulatú bőrrák detektálása 3D-TBP segítségével
(December 2024)

Attila Nemes (B6RYIK), Csaba Potyok (OZNVQ4) and Peter Arany (U4VQHM)
IsIcTeam2024

***Absztrakt* — A beszámolónkban a féléves projektünk során a bőrrák felismerésére általunk használt modelleket és módszereket ismertetjük. A bőrrák felismerését gépi látási problémaként közelítettük meg, amelyhez a MobileNet-v3 [1], az AlexNet és egy saját konvolúciós neurális háló alapú modellt használtunk. A modelleket kombinálva is kipróbáltuk az eredmények javításáért, valamint keresztvalidációval visszaellenőriztük a mért eredmények megbízhatóságát. A modellek teljesítményét a pAUC metrika alapján hasonlítottuk össze, hogy lássuk, melyik modell lenne eredményesebb a versenyen. A képfeldolgozáshoz egy speciális szőr eltávolító algoritmust is kipróbáltunk, amelyet egy cikk alapján valósítottunk meg. Többféle augmentálási módszert használtunk a tanítóadathalmaz gazdagítására, ami azért volt fontos, mert az adathalmazban kevés olyan minta volt, amely rosszindulatú bőrelváltozást tartalmazott. Több modellel is kiértékeltük a tesztadathalmazunkat és összeségében sikerült 15.9 pAUC értéket elérnünk a legjobb modellünkkel. Legvégül készítettünk egy AI szolgáltatást, amely a modellünket használja a feltöltött képek osztályozására.**

***Abstract* — In this report we describe the models and methods we have used to detect skin cancer during our semester-long project. We have approached skin cancer detection as a computer vision problem, using MobileNet-v3 [1], AlexNet and a proprietary convolutional neural network based model. We also tested the models in combination to improve the results and cross-validated the reliability of the measured results. The performance of the models was compared using the pAUC metric to see which model would be more effective in the competition. For image processing, we tried a special hair removal algorithm, which was implemented based on an article. We used multiple augmentation methods to enrich the training dataset, which was important because there were few samples in the dataset that contained malignant skin lesions. We evaluated our test dataset with several models and overall we achieved a pAUC of 15.9 with our best model. Finally, we compiled an AI service that uses our model to classify the uploaded images.**

***Index Terms* — Computer vision, Convolutional neural network, Hair removal, Image augmentation, pAUC metric, Skin cancer, Skin malignancy.**

I. Introduction

Skin cancer can be deadly if not caught early, but many populations lack specialized dermatologic care. Over the past several years, dermoscopy-based AI algorithms have been shown to benefit clinicians in diagnosing melanoma, basal cell, and squamous cell carcinoma. However, determining which individuals should see a clinician in the first place has great potential impact. Triaging applications have a significant potential to benefit underserved populations and improve early skin cancer detection, the key factor in long-term patient outcomes.

Dermatoscope images reveal morphologic features not visible to the naked eye, but these images are typically only captured in dermatology clinics. Algorithms that benefit people in primary care or non-clinical settings must be adept to evaluating lower quality images. We used the International Skin Imaging Collaboration (ISIC) 3D TBP dataset, which was introduced in the Kaggle competition [2]. It is a new dataset that included all the lesions of thousands of patients on three continents, with images resembling mobile phone photos. However, the drawback of the dataset is that it is unbalanced, as positive samples for skin cancer accounted for 0.097% of the total dataset.

II. Related Works

The following articles and works were used throughout the project. They helped us to process the data, learn about models, use and create them.

A. Unbalanced Data

To deal with unbalanced data sets, we found the following two articles. (M. Mayo, 2024) [3] This article discusses what metrics and methods can be applied in general to an unbalanced dataset. These are data resampling, algorithmic ensemble methods, adjusting class weights, using appropriate evaluation metrics and generating synthetic samples. Among these, we used data enrichment and class weighting. (Yun-Chun Wang, Ching-Hsue Cheng, 2021) [4] This work presents a concrete data processing workflow that the authors of the article used to achieve good results in the classification process.

B. Skin cancer classification

We also used articles on the classification methods used to identify skin cancer. There were also two that provided an overview report of what some models have achieved so far in other projects. (Maryam N, Syed Q. G. etal., 2023) [5] In this citation the AlexNet, ResNet, VGG, MobileNet, DenseNet models and their improved versions for classification were discussed. These models were then compared with each other based on their performance. This shows that AlexNet achieved

98.33%, MobileNetV2 98.2%, ResNet101 and DenseNet201 98.7% accuracy. (Yinhao W., Bin C. etal., 2022) [6] This work though is stating that these models achieved other results. This could be due to the fact that this article is one year earlier than the first one. (Hosny, K.M., Kassem, M.A. & Fouad, M.M., 2020) [7] In this article, the AlexNet network was transfer trained that achieved an outstanding result on the 2018 ISIC dataset, classifying the skin lesions in the images into not just 2 but 7 classes. The achieved results are 98.70%, 95.60%, 99.27%, and 95.06% for accuracy, sensitivity, specificity, and precision, respectively.

(Rupali K. S., Md. Shahinur A., etal., 2022) [8] This work talks about a model that can run on low power machines. However, this article was also interesting for us because it presented a hair removal algorithm that could significantly improve the accuracy of the model.

(Esteva, A., Kuprel, B., Novoa, R. etal., 2017) [9] This article presents the construction of a proprietary CNN model for the detection of malignant skin lesions. The accuracy of the solution is compared with the expertise of dermatologists and it is shown that the model achieves this knowledge.

C. Data preprocessing

(Viknesh, C. K., Kumar, P. N., Seetharaman, R., & Anitha, D., 2023) [10] This paper discusses how different image processing techniques can be used to improve the accuracy of models on ISIC datasets. It describes in detail these techniques and how this has affected the results of the models.

D. Data enrichment

(Thomas W. Sederberg, Scott R. Parry, 1986) [11] This work talks about a data enrichment method that uses non-affine transformation to generate new images. The essence of this is that the image is filtered as if a canvas were being folded on a flat surface.

E. Systematic method for malignant detection

(Ebrahim Mohammed Senan, Mukti E Jadhav, 2021) [12] This article shows the so-called ABCD method, which can be used to detect systematic malignant lesions. The acronym is derived from: asymmetry, borders, colours and diameter. It demonstrates how the method is composed of elements and how it performs as a fundamentally non-artificial intelligence based solution on a given test dataset.

III. DATA PROCESSING

At the beginning of the work, we collected the ISIC 2024 dataset made available at the Kaggle competition. For efficient work we also uploaded it to a dedicated DropBox repository so that team members could easily access and use it during the modelling. The dataset contains every lesion from a subset of thousands of patients seen between the years 2015 and 2024 across nine institutions and three continents.

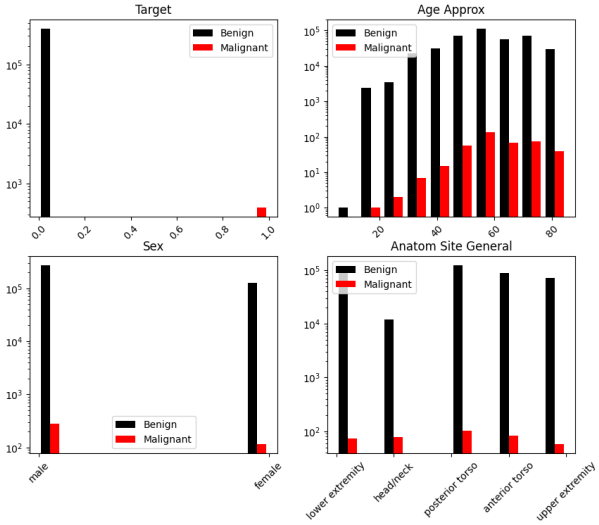
A. Features

The dataset consists of diagnostically labelled images with additional metadata. The images are JPEGs. The associated .csv file contains a binary diagnostic label (target), potential input variables (e.g. age_approx, sex, anatom_site_general, etc.), and additional attributes (e.g. image source and precise diagnosis). To mimic non-dermoscopic images, the Kaggle competition uses standardized cropped lesion-images of lesions from 3D Total Body Photography (TBP).

The 1. Figure shows 9 images taken from the training dataset, including images of malignant lesions. The dimensions of each image vary, but at least it is guaranteed that the images have been properly cropped. The skin areas in the images also contain hair, which in some cases obscures interesting details of the lesion.

1. Figure Images of the part of the dataset showing a malignant lesion.

Each image is associated with a 55-dimensional feature vector, which also contains the value of the target. This includes data that is not filled in for most of the images and is difficult to fill in properly afterwards. Therefor these feature vectors were discarded from the dataset during processing. We also examined the distribution of the tabular data for the columns 'target', 'age_approx', 'sex' and 'anatom_site_general'. With this we wanted to see how the possible values of the given columns were distributed in the training dataset. Doing so, we noticed a significant shift towards benign lesions in the images in the dataset.



2. Figure Distribution of values for these features.

The training dataset contains a total of 401059 images, of which only 393 contain malignant lesions. This is such a negligible proportion that we had to plot the distribution of the two classes on a logarithmic scale on the above diagram.

B. Difficulties

Our first difficulty with the dataset was that only a 3-item test dataset was made available during the competition, and the expected target values were missing, so we could not use them in the model evaluation. Therefore, we had to re-divide the training dataset into 3 parts (training, validation and test sets), from which we separated a test dataset with 75-75 positive and negative samples.

The second big problem was the imbalanced data set mentioned earlier, where there were still a few orders of magnitude more negative samples than positive ones. We approached the problem from several directions, taking into account the solutions mentioned in previous articles. As part of this, we decided to have an equal proportion of positive and negative samples in our test dataset, but we also had to be careful not to overcommit the test dataset and thus not leave enough positive samples in the training and validation datasets.

Finally, the size of the validation set was chosen to be 75-75 and the remaining image was used for training. However, class balancing in the training data set was solved by resampling. The negative samples were significantly undersampled, while the positive samples were oversampled in order to improve their proportion in the training dataset. However, we introduced augmentation methods for the training set in order to increase the generalisation ability of our models.

C. Methods of data preprocessing

In addition to images, we also processed tabular data during the project. One of our approaches used them as input variables for our models. From the available columns, we selected some, included both numeric and categorical variables. The missing values were filled in using the ffill() method of pandas, fortunately for our chosen columns there were only minimal missing values, so we were able to correct with this solution.

Among the categorical variables, the columns "sex", "anatom_site_general", "tbp_tile_type" and "tbp_lv_location" were used and their values were one-hot encoded using the get_dummies method of pandas to ensure that the model does not learn the relative value of each value incorrectly, but treats them as equal. The numeric variables are "age_approx" "tbp_lv_nevi_confidence", "clin_size_long_diam_mm", "tbp_lv_areaMM2", "tbp_lv_area_perim_ratio", "tbp_lv_color_std_mean", "tbp_lv_deltaLBnorm" and "tbp_lv_minorAxisMM" columns were used, standardized by column based on the training set.

D. Data augmentation and hair removing

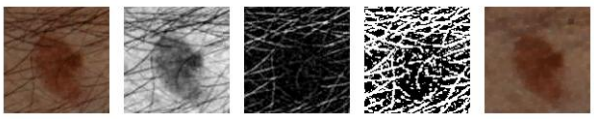
To enrich the samples in the training dataset, different augmentation transformations were applied randomly to the images in the training data set, no such transformations were applied to the tabular data. For the transformations we used the torchvision transforms functions RandomHorizontalFlip, RandomVerticalFlip, RandomRotation and RandomErasing. We also wanted to use ElasticTransform, but it slowed down the learning significantly and so it was eventually abandoned. Nevertheless, we also tried to create our own non-affine transformation, but in the end, we did not use it.

Since the details in the images were not sensitive to orientation, we found it perfectly legitimate to use different rotations and reflections, but we did not use random cropping and zooming because it might have thrown out important parts of the image (although this would have been negligible). Finally, the images were transformed into tensors. The tensor values from the colour images were transformed from the range [0..1] to the range [-1..1].

While reading the review article, our eyes were caught by the hair removal algorithm [8]. This can be use to remove hairs from images without significantly distorting the details of the image. To implement the algorithm, we looked at the original article, which described a solution used, but did not go into much detail. The algorithm is based on the OpenCV methods, which can be used to remove thin hairs from the image by applying them in the right order.

The Squeeze algorithm steps:

1. Get structuring elements
2. Convert the image to gray
3. Making morphological transformation
4. Making a mask from the details to be removed
5. Re-painting of masked areas



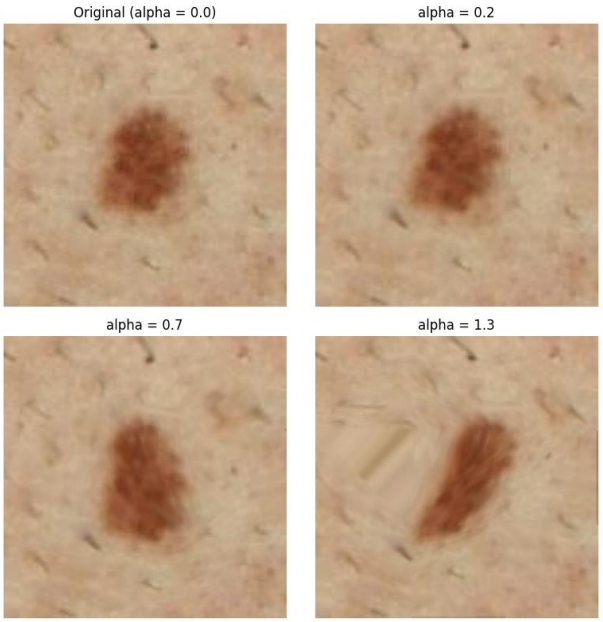
3. Figure Example for Squeeze algorithm usage.

E. Data augmentation: Free-Form Deformation based image transformation

The method's operation is straightforward. First, a grid must be defined on the original image. In our case, we opted for the simplest approach, creating a uniformly distributed NxN grid, where N typically ranges between 4 and 6. One drawback of this deformation is that it blurs the image when a specific region is stretched. Therefore, a more optimized grid allocation method could also be considered.

In the next step, we associate the pixels of the image with the grid points (a pixel may be influenced by multiple grid points). Following this, the grid points are displaced by a degree dependent on a parameter α. Finally, the pixels are transformed based on the new positions of the grid points affecting them.

Based on our observations, non-affine transformations significantly slow down training times, and their added value is minimal compared to affine transformations and other augmentation techniques. As a result, we generally did not use them.



4. Figure Illustrates the results for different α values on a test image.

IV. Model architectures

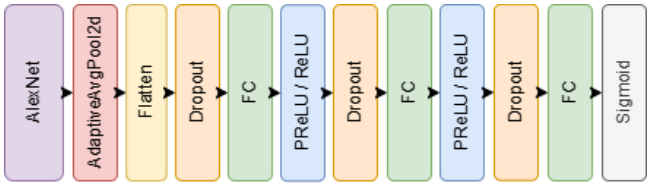
At the start of the modelling, we created a simple baseline model, which we taught for a few epochs. Based on the articles we read, we tried to model the problem along 3 tracks. One way was to use existing models, replace their classifier layers with own custom classifier layer and train the pre-trained nets with this new classifier layer. As another option, we used tabular data in addition to images and processed the images with a pre-trained network and then combined the results of the two. Finally, we created a model combination that evaluates the images using two pre-trained networks simultaneously, and then a selector network learns to weight the results of the two models.

A. Baseline model

The baseline model was built from a 1-layer Linear layer, which we tried to build a bit like the LogisticRegression model to make it as simple as possible. The model was obtained by first converting the images to a lower resolution using an AvgPool2d kernel, and then expanding them to 1-dimension using a simple Flatten operation. The stride was set to 8 for AvgPool2d, so the images were compressed to 1/64 by the model before the Linear layer received them as input.

B. AlexNet-based model

In the review article, we found that as a first attempt, it might be worth using the AlexNet to solve the problem. This model was mentioned in the article as the first solution that has been able to produce acceptable results in this area. The pre-learned AlexNet was trained for the ImageNet task, so it is not a one-to-one match for us. Therefore, we kept only the features layer of the network and replaced it with our own fully-connected layer, which correctly solves our classification problem.

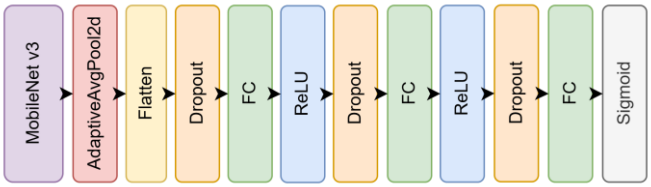


5. Figure The architecture of AlexNet-based model.

The 5. Figure shows the architecture of the AlexNet based model, where after the last FC layer a sigmoid activation function maps the model output to probabilities for classification. For the AlexNet-based model, we also tested the use of ReLU and PReLU activation layers. Overall, no major differences were found when using these activation functions.

C. MobileNetv3-based model

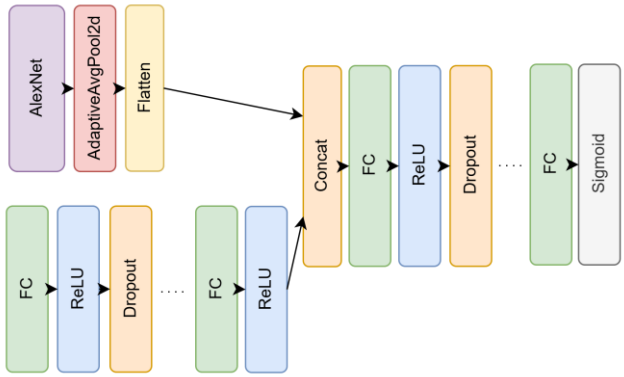
As a next step, we tried the MobileNet v3 model based on the review article, after which we built a classification network with a similar structure to the AlexNet-based model using FC layers. We used the small version of the MobileNet v3 model and started from the pre- trained network for the training.



6. Figure The architecture of MobileNetv3-based model.

D. Complex model for images and tabular data

We also created a model that uses the metadata associated with the images. The model receives images and tabular data simultaneously, processing the images using AlexNet and processing the tabular data using FC layers. The output from the images and tabular data is concatenated, on which a multilayer FC net performs the classification.



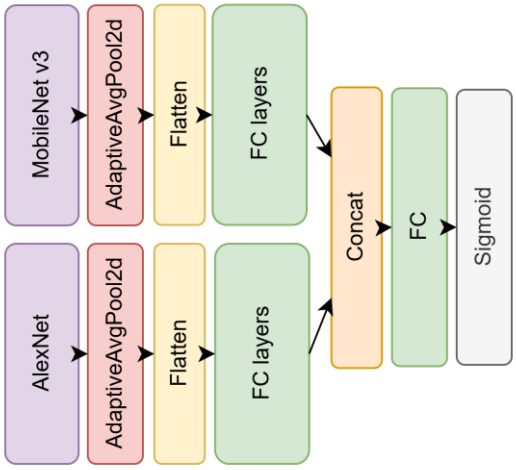
7. Figure The architecture of the model for images and tabular data.

The dotted parts on the 7. Figure indicate that similar layers follow each other, with the difference that the number of neurons in each layer varies.

E. Combined model with selector network

The previously created AlexNet and MobileNet v3 based models are combined with a simple model selector network, where the output of the two models is concatenated and then used to produce a prediction using a small FC layer. The last FC layer forms the model selector part, which is responsible for learning how to weight the values for each model output. It is of course possible to use a more complex model selector net, but we have not done so here.

Both pre-trained nets receive the same input to produce their own representation. The same classification networks were used for the two models except for the sigmoid activation function. The model selector FC layer receives the output of the last FC layer of the models concatenated. To obtain the final shape, we also used a sigmoid activation function.



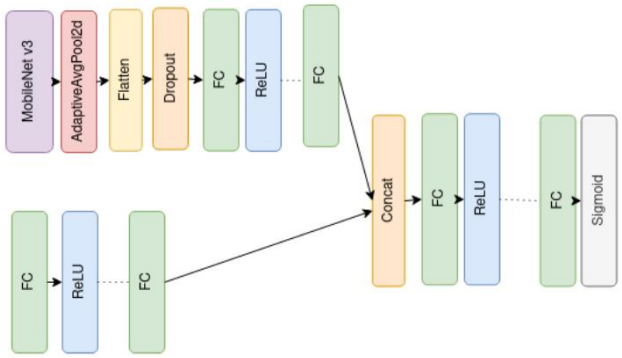
8. Figure The architecture of combined models.

F. Simple image-based model with simple metadata-based model

We conducted a simple experiment to examine what happens when we take an image-based model and a metadata-based model, train them separately, and then combine their predictions during evaluation by averaging the outputs. The method is primitive but easy to test, as the training process produces two independently testable models. By evaluating them together, we can directly compare their combined performance against the baseline models.

G. Complex model for images and tabular data (MobilNetv3)

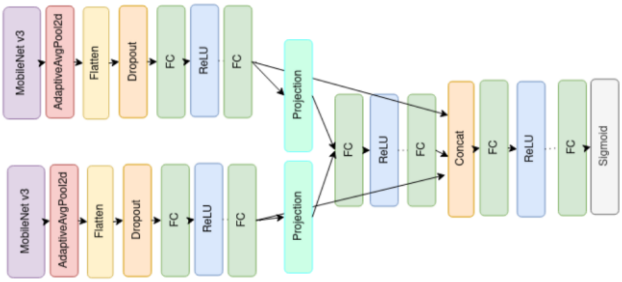
Similar to training AlexNet alongside tabular data, we also experimented with using a pre-trained MobileNetv3 network instead of AlexNet for image-based training to evaluate the achievable results.



9. Figure The architecture MobileNet-based complex model.

H. Combined image-based model with tabular data-based model

Finally, we tested a slightly more complex architecture. The model incorporates an AlexNet, a MobileNetv3, and a fully connected (FC) metadata-based model. The image-based models directly process the images, while the metadata-based model receives the outputs of the two image-based models as additional inputs. However, a projection layer first reduces the number of neurons in these outputs to ensure the metadata-based model does not overly rely on them, using them only as supplementary information.

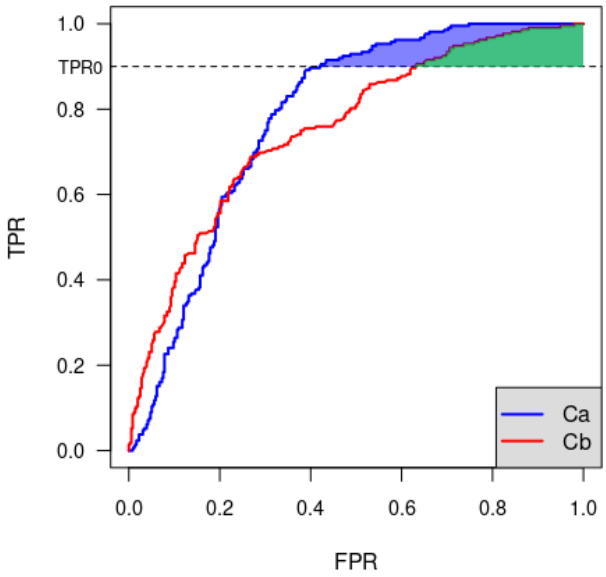


10. Figure The architecture of the combined model with tabular data

V. METRICS TO EVALUATE MODELS

When evaluating the models, accuracy was not the most important metric we used to compare the models' capabilities. The reason for this was that it was not used as the main criterion in the articles on the topic, nor was it the basis for comparison in the Kaggle competition. Several metrics were measured when comparing the models: f1-score, recall, precision and AUC. Detection of malignant skin lesions is key, so the recall value is a very important metric for comparing the models, as it shows the proportion of samples with truly malignant skin lesions that the model found. The AUC value can be used to determine how well our model classifies. The Kaggle competition used partial AUC to compare models.

"The receiver operating characteristic (ROC) curve illustrates the diagnostic ability of a given binary classifier system as its discrimination threshold is varied. However, there are regions in the ROC space where the values of TPR are unacceptable in clinical practice. Systems that aid in diagnosing cancers are required to be highly-sensitive, so this metric focuses on the area under the ROC curve AND above 80% TRP. Hence, scores range from [0.0, 0.2]."- about pAUC from the Kaggle competition.



11. Figure Illustration of pAUC from the Kaggle competition site.

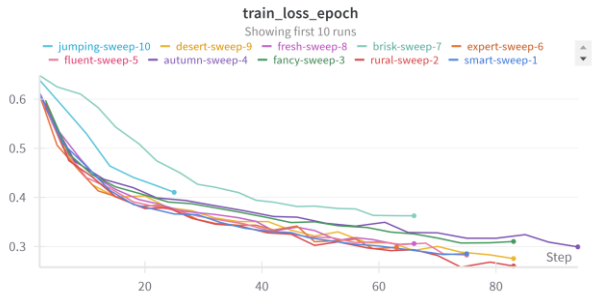
The pAUC metric is based on the implementation on Kaggle and implemented using the torchmetrics package. For the pAUC value, the lower threshold for TPR was set at 80%, as for the competition. Confusion matrices were also created for the test results to make it easy to understand how the quantified metrics performed as they did.

VI. METHODS OF TRAINING MODELS

The models were built using Pytorch and the training, validation and test pipelines were implemented using Pytorch Lightning. We defined a single LightningModule, which got the models in its parameter and performed the single process (training, validation, testing and evaluation). Since a classification task is required to solve the models, we computed and optimized BinaryCrossEntropy (BCE) loss during training. During backpropagation, SGD was used to optimise the loss. During the training, the best models were always saved using ModelCheckpoint, and EarlyStopping was used to stop the training early if the desired metric was no longer improved within a given epoch number.

Before training, we split the data set as mentioned earlier and resampled the training data set sample to rebalance the dataset. Positive samples were oversampled by a factor of 6, while negative samples were undersampled by a factor of 0.0075. The data was processed in mini-batches using Dataloader. The images were dynamically loaded by the Dataloader from hdf5 format file, and then augmentation transformations were performed on the training images.

Model training was done via Colab, where we logged the model training and validation steps using Tensorboard and Wandb to find the best models. Since the models we used did not have many parameters (only a few million), we did not need to look for other running environments. The 12. Figure shows the train loss produced by one of the hyperparameter optimizations over time, where it can be seen that models learn nicely and the loss decreases as the epochs progress.

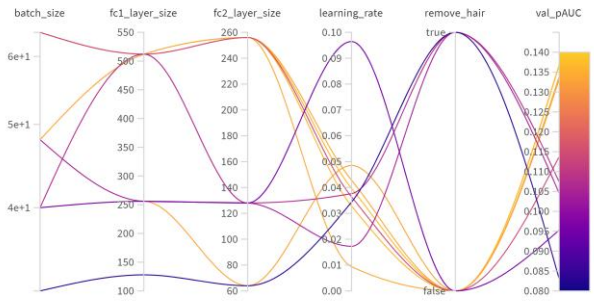


12. Figure Evolution of train loss during model learning.

VII. HYPERPARAMETER OPTIMIZATION

In addition to manual hyperparameter optimization, we also implemented automated optimization using the Wandb framework. During the parameter optimization, we selected some parameters to be optimized, such as learning_rate, number of neurons in the classifier FC layers, batch size and whether to apply Squeeze algorithm during image processing. For the optimizations, we tried both random and Bayesian hyperparameter optimization. The best results were obtained with the Bayesian optimization.

Separate hyperparameter optimizations were performed on the AlexNet and MobileNetv3 based models to see which architecture performs better on this dataset. The results were similar for the models with minimal differences. During the optimization, we found that there are some configurations where the squeeze algorithm is able to improve the model's performance, but we were able to achieve similarly good results by reconfiguring other parameters.



13. Figure Diagram of parallel cordinate systems for AlexNet based model optimization.

During the hyperparameter optimization, we found that the learning_rate setting mattered the most, but the number of neurons in the classifier FC layers also affected the result. During the optimization, we compared the models primarily based on the pAUC value measured on the validation dataset, as this was the main basis for comparing the models in the competition.

VIII. Results

A. Evaluation of models

The best models after hyperparameter optimization were also evaluated on our separate test dataset. Here again, the pAUC of the models was the primary comparison, but we also examined the results obtained for the other calculated metrics.

The 1. Table shows the best possible results for the models we used (here we have included the results from cross-validation). We have highlighted the values that were the best for the given metric. Based on these results, the MobileNetv3-based model performed best for us in terms of pAUC, but this does not guarantee that the best possible results were found for the other models.

Model	pAUC	recall	precision	f1-score	loss
Baseline	*0.077*	*0.573*	*0.860*	*0.688*	*0.574*
AlexNet based	0.116	0.800	0.822	0.811	**0.434**
MobileNetv3 based	**0.159**	0.760	**0.950**	0.844	0.445
Combined (AlexNet + MobileNet v3)	0.134	**0.840**	0.900	**0.869**	0.455
Complex (images + tabular data)	0.125	**0.840**	0.851	0.846	0.446
Complex (images + tabular data MobilNetv3)	0.151	0.600	0.918	0.725	0.503
Combined + tabular data	0.086	0.773	0.734	0.753	0.537

1. Table The results of the models on test dataset.

You can also see that we managed to beat our baseline model, although it was very rudimentary. The more complex solutions did not perform badly either as their recall was higher than the MobileNetv3 based model with the best pAUC. During testing we often found that the best pAUC values are not always accompanied by a high recall value. The reason for this was seen to be that the models tended to learn which samples were the ones that contained benign lesions, which in our case was less advantageous.

In this regard, the models with higher recall values seem to be better able to find the highest number of malignant lesions and more in the opposite sense. In our opinion, it is better to detect a benign lesion as malignant than vice versa, and with medical expert knowledge it is possible to override a lesion that was mistakenly considered malignant afterwards, the delay of which is still not as much of a problem as in the reverse case.

B. Cross-validation

As we selected the new training, validation and test dataset from the original training dataset completely at random, without any prior medical expertise, we thought it was important to examine the uncertainty introduced into the results. The test dataset remained the same as the 150-item sample set selected previously, which was left unchanged throughout. However, the training and validation sets were recreated during every single cross-validation run.

In preparation for cross validation, we split our data set into 75-75 size parts along the positive samples excluded from the test dataset, where we re-sampled 75 positive and 75 negative samples into a validation dataset. The remaining samples were then resampled as described above. We then trained the models for 20 epochs on the given training and validation dataset, using the best hyperparameters for training and extracting the best model after each epoch. We found that the results are close to each other, but there is still a small variation between model results. We also evaluated the model produced at the end of each run on the test dataset, in order to see how the test results would have evolved if we had applied the given training and validation dataset decomposition.

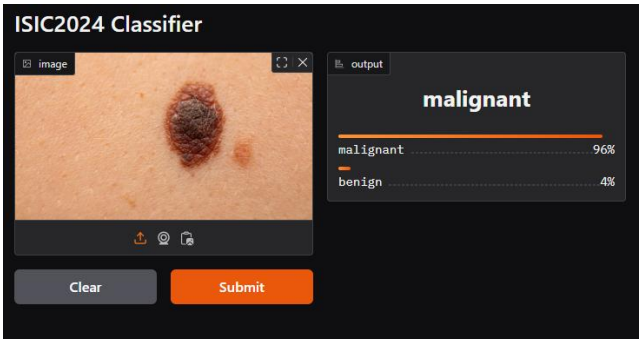
Model	Run 1	Run 2	Run 3	Run 4	Run 5
AlexNet based	0.115	0.110	0.077	0.112	0.116
MobileNetv3 based	0.150	0.159	0.149	0.140	0.143
Combined (AlexNet + MobileNet v3)	0.124	0.103	0.101	0.134	0.138
Complex (images + tabular data)	0.128	0.126	0.127	0.125	0.126
Complex (images + tabular data MobilNetv3)	0.153	0.148	0.078	0.089	0.109
Combined + tabular data	0.098	0.146	0.089	0.145	0.094

2. Table The pAUC of the given cross-validation run on test dataset.

Overall, it can be concluded that there was no significant negative effect of randomly selecting a validation set with a fixed test data set. Only the combined models and MobileNetv3-based complex model showed larger fluctuations, in the other cases the results were stable in terms of pAUC. Of course, it would be worth comparing the other values in this way, because, for example, the recall value also varied quite a lot during each cross-validation run.

IX. Building AI service

In order to use the project as a service, we created a gradio service from one of the best models. This service can be run locally in a docker container (to run it on localhost, additional instructions can be found in the README.md file in the project's github repository). A static version of the service was also loaded on the Huggingface site (one of the models is fixed and used in the service), so the service is available online. Currently, the space is publicly available at the link [13] but has the limitations of a free service. This is mainly related to the size of the model (16 GB) and the computing resource (CPU), which does not affect us.



14. Figure Our working AI service on Huggingface.

X. Future plans

The project could be developed in many more ways in the future. We were thinking about expanding the dataset, improving the accuracy of models and testing other models.

We thought about expanding the dataset because the more data a network learns on, the better it will perform its task. And for us it would be important to have skin cancer positive samples in at least 50% of the data set. This desired limit could be achieved by generating data with generative models or by collecting data from a reliable source.

By improving the accuracy of our models, we meant that we should achieve better pAUC values with our models. This could possibly be achieved by further hyperparameter optimizations or by using additional promising methods.

There are some models that we have not tried, but others have achieved good results in skin cancer detection. Of these models, we think that it would be worth trying the DenseNet, ResNet and VGG models. Of course, combinations of the models we have used or listed above would also be worth trying.

XI. Conclusion

Throughout our semester-long project, we have learned a lot about computer vision and have become more familiar with its use for medical purposes. We also learned about the advantages and difficulties of using it. We consider the results of our models to be successful, although we could not come close to the results achieved in the competition, but as we did not have the opportunity to evaluate our models on the data that was used in the competition, we could not compare them perfectly with the data used in the measurements. Of course, we consider that there is room for improvement in the results, but you cannot fit this into the semester timetable. We are pleased that we were able to make our modelling results available and testable in the form of an AI-based service.

References

[1] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). *MobileNets: Efficient convolutional neural networks for mobile vision applications*. arXiv. https://arxiv.org/abs/1704.04861

[2] Kaggle competition, ISIC 2024 - Skin Cancer Detection with 3D-TBP | Kaggle

[3] M. Mayo, "5 Effective way to Handle Inbalanced Data in Machine Learning", Jun. 26. 2024, [Online]. Available: https://machinelearningmastery.com/5-effective-ways-to-handle-imbalanced-data-in-machine-learning/

[4] Yun-Chun Wang, Ching-Hsue Cheng, A multiple combined method for rebalancing medical data with class imbalances, Computers in Biology and Medicine, Volume 134, 2021, 104527, ISSN 0010-4825, https://doi.org/10.1016/j.compbiomed.2021.104527

[5] Naqvi, M., Gilani, S. Q., Syed, T., Marques, O., & Kim, H. C. (2023). Skin Cancer Detection Using Deep Learning-A Review. Diagnostics (Basel, Switzerland), 13(11), 1911. https://doi.org/10.3390/diagnostics13111911

[6] Wu, Y., Chen, B., Zeng, A., Pan, D., Wang, R., & Zhao, S. (2022). Skin Cancer Classification With Deep Learning: A

Systematic Review. *Frontiers in oncology*, *12*, 893972. https://doi.org/10.3389/fonc.2022.893972

[7] Hosny, K.M., Kassem, M.A. & Fouad, M.M. Classification of Skin Lesions into Seven Classes Using Transfer Learning with AlexNet. *J Digit Imaging* **33**, 1325–1334 (2020). https://doi.org/10.1007/s10278-020-00371-9

[8] Shinde, R. K., Alam, M. S., Hossain, M. B., Md Imtiaz, S., Kim, J., Padwal, A. A., & Kim, N. (2023). Squeeze-MNet: Precise Skin Cancer Detection Model for Low Computing IoT Devices Using Transfer Learning. *Cancers*, *15*(1), 12. https://doi.org/10.3390/cancers15010012

[9] Esteva, A., Kuprel, B., Novoa, R. *et al*. Dermatologist-level classification of skin cancer with deep neural networks. *Nature* 542, 115–118 (2017). https://doi.org/10.1038/nature21056

[10] Viknesh, C. K., Kumar, P. N., Seetharaman, R., & Anitha, D. (2023). Detection and Classification of Melanoma Skin Cancer Using Image Processing Technique. *Diagnostics (Basel, Switzerland)*, *13*(21), 3313. https://doi.org/10.3390/diagnostics13213313

[11] Thomas W. Sederberg and Scott R. Parry. 1986. Free-form deformation of solid geometric models. SIGGRAPH Comput. Graph. 20, 4 (Aug. 1986), 151–160. https://doi.org/10.1145/15886.15903

[12] Ebrahim Mohammed Senan, Mukti E Jadhav, Analysis of dermoscopy images by using ABCD rule for early detection of skin cancer, Global Transitions Proceedings, Volume 2, Issue 1, 2021, Pages 1-7, ISSN 2666-285X, https://doi.org/10.1016/j.gltp.2021.01.001

[13] The online service of our project, https://huggingface.co/spaces/Nemes2000/isic_2024

An English version of this documentation was produced using DeepL.