ORO Santé/ORO Health

Skin Disease Classification Documentation

Author: *Shanel Gauthier*

1 Dataset Description

The task consists of classifying three skin disease types from skin images. The three classes are “acne”, “herpes simplex”, and “lichen planus”. Samples of the dataset are shown in Figure 1. In total, there are 102 images: 40 images of acne, 16 images of herpes simplex and 46 of lichen planus, as shown in the Figure 2. The images have different sizes. The minimum and maximum values

Figure 1: Sample images of the dataset

for the width are 298 and 1200 respectively. The minimum and maximum height values are 298 and 1500, respectively. The histograms in Figure 3 show the distribution of the widths and heights of all the images in the dataset.

There are three key challenges associated with this dataset.

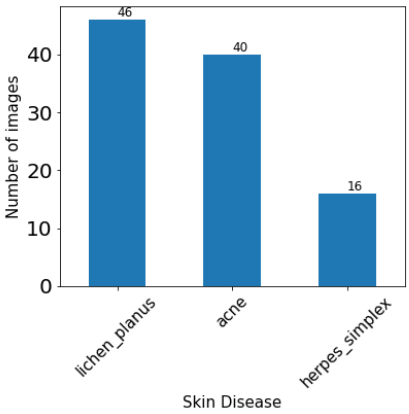


Figure 2: Class Distribution

1. **Challenge 1: Learning from an imbalance dataset.** The dataset is imbalanced because there are only 16 images of herpes simplex. We need to take this information into account during the training of the different models.

2. **Challenge 2: Efficient learning from limited labeled data.** One of the key challenges here from a machine-learning perspective is a limited amount of labeled data. Learning useful representations from little training data is arduous. The model can easily overfit the small training set.

3. **Challenge 3: Finding disentangled representations.** A crucial element in extracting information features from high dimensional structured data is the disentanglement of sources of variations in the data. We observe a lot of intra-class variation in Figure 1. For example, within the same class, the images contain a different part of the body. The luminosity is different

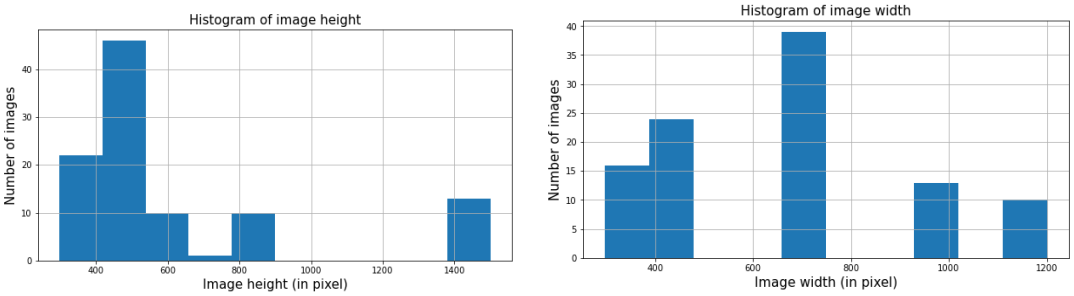


Figure 3: Histogram of the (left) widths and (right) heights of all the images in the dataset.

between all the images. We can consider global translations, rotations and scale as being caused by the positioning of the camera.

2 Assumptions

We list all the assumptions made for this experiment.

- We assume that all the images in the dataset come from different patients. In others words, we assume that there are 102 images associated with 102 patients. We do not want to have images from the same patient in the train and test set.
- The model will be tested against a separate set of images with the same disease types. We assume that the separate set of images has a similar distribution to the dataset that we are using for this experiment. That being said, we assume that the second dataset is also imbalanced (e.i., contains fewer images of herpes simplex).

3 Architectures

We consider four different architectures for this task. In this section, we describe the architectures.

3.1 Vanilla CNN

We consider a vanilla convolutional neural network (CNN) as the baseline of the work. Convolutional Neural Network (CNN), also called ConvNet, is a deep learning algorithm that takes the spatial information of images into account. In a CNN, hidden units are connected to sub-regions of the image and are also connected to all channels, thus reducing the number of parameters. CNN utilizes the concept of parameters sharing with kernels (also called filters) that slide along the input image, hence further reducing the number of parameters. As shown in Figure 4, typical CNNs are made up of three different layers: the convolutional layer, the pooling layer, and the fully-connected layer. In this work, we consider a 3-layers vanilla CNN with 1M trainable parameters.

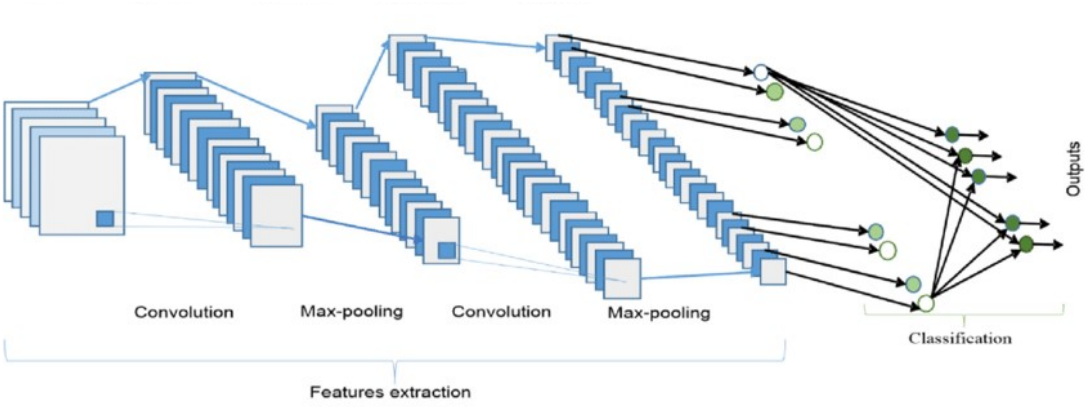


Figure 4: **CNN Architecture.** Typical CNNs are built of convolutional layers, pooling layers and fully-connected layers. In the convolutional layer, we slide the kernels across the input matrices. The outputs of the convolutional layer are the features maps. Image from:[1].

3.2 Petrained Resnet-50

In this work, we also consider the use of transfer learning. Transfer learning was built on this idea of transferring features across tasks. Transfer learning consists of training a network on a different dataset. Then, the learned weights are used as the initial weights on a different classification task. A lot of attention in the research community has been put on using natural images to study transfer learning onto the domain of medical imaging [2, 3, 8, 9, 10, 12]. We consider the Resnet-50 pretrained on ImageNet since it is quite common to use this pretrained model on medical images. During the training, we unfreeze all the weights in the pretrained network. The pre-trained weights are used as initial weights but are optimized with respect to the new task. The total number of trainable parameters is 23M.

3.3 BiT

The third consider architecture is called Big Transfer (BiT) [5] and was introduced in 2019. BiT is also a transfer learning approach that considers numerous improvements to deep learning that have been introduced recently. It is pretrained on a large supervised source dataset and finetuned on the target task. The network is high-capacity. BiT yields state-of-the-art results in the small data regime. In this work, the total number of trainable parameters is 42M.

3.4 Parametric Scattering Network

The last architecture considered is the parametric scattering network [4], which has shown to be useful in the small data regime. I am the first author of this paper [4]. The approach is built on the scattering network. Scattering network, proposed in [6], is a cascade of wavelets and complex modulus nonlinearities, which can be seen as a convolutional neural network (CNN) with fixed, predetermined filters. The scattering construction is based on complex wavelets, generated from a mother wavelet via dilations and rotations. Further, discrete parameterization and indexing of these operations have traditionally been carefully constructed to ensure the resulting filter bank forms an efficient tight frame with well-established energy preservation properties. In the parametric scattering network, we relax the standard wavelet tight frame construction by considering another alternative where a small number of wavelet parameters used to create the wavelet filterbanks are optimized for the task at hand. In other words, focusing on Morlet wavelets, we learn the scales, orientations, and aspect ratios of the filters to produce problem-specific parameterizations of the scattering transform.

In this work, for classification purpose, we combine the parametric scattering network with a linear layer, as shown in Figure 5. In total, the network has 7M trainable parameters.

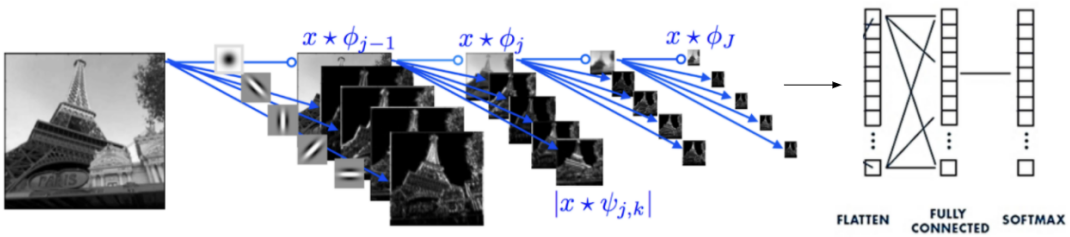


Figure 5: Example of the parametric scattering network followed by a linear layer. Image adapted from: [7].

4 Implementation Details

In this section, we describe the hardware, software information and implementations details.

Machine Learning Framework. The models are implemented using the Py-

Torch open-source machine learning framework.

Hardware. The experiments leverage a TITAN RTX GPU with CUDA version 11.2 and 24GB of memory. The operating system is Linux.

Software. All the python packages needed to run the experiments are listed in the *environment.yml* file. The experiments are run using 4 different Jupyter notebooks.

Deterministic Results. To obtain comparable and reproducible results, we control for deterministic GPU behaviour. We use the same set of seeds for models evaluated on the same number of samples.

Data Augmentation. In the experiments, the training set is only augmented with horizontal flipping.

Loss Function. The classification is done via a softmax layer yielding the final output. All models are trained using cross-entropy loss, minimized by stochastic gradient descent. Since the dataset is imbalanced, the loss function was modified to take into account the imbalanced dataset. The CrossEntropyLoss() function takes an argument called "weight" that allows the user to define values to the importance to apply to each class. We increase the weight of the minority class so that its loss also increases. This forces the model to learn from the minority class.

Cross-Validation. To evaluate the models, we use 5-folds cross-validation. The folds are stratified.

Normalization. We compute the mean and standard deviation per channel of the dataset. For this dataset, the mean per channel is : [0.6475, 0.4907, 0.4165] and the standard deviation per channel is [0.1875, 0.1598, 0.1460]. Then, each image is normalized per channel by the mean and standard deviation.

Metrics. For each fold, we save the classification report and then do an average over the 5 folds to evaluate the models. Between all the metrics saved in the classification report, we consider recall the most important metric. In medical applications, recall is often considered one of the most important because it is desirable to miss as few positive instances as possible. The most important metric is the average macro recall in this study since the 'macro recall' calculates metrics for each disease and computes the average.

Resizing. Since the images have different sizes, the images are resized to 400 pixels x 400 pixels. Random Copping and center cropping were also considered. However, randomly cropping areas of images may miss the affected region of the body. Also, the affected area is not necessary in the center of the image.

Figure 6: Examples of random cropping. Randomly cropping areas of images may miss the affected region of the body.

5 Results

We evaluate the performance of the four architectures on this task. For each architecture, we perform a simple hyperparameter search to find the best combinations of mini-batch size, learning rate and the number of epochs. All the results of the hyperparameter searches are saved using the MLflow experimental tracking tool. We saved the average values for each metric over the 5 folds. The metrics are saved in the DAGsHub's servers. The results are available on `https://dagshub.com/gauthier.shanel/skin_disease/experiments/`. The top hyperparameter searches results are shown in Figure 7.

model	total_...	batch_size	lr	trainable_parameters	num_epochs	macro avg-recall-avg ↓	macro avg-precision-avg	macro avg-f1-score-avg
bit	424.212...	4	0.0031	42498627	22	0.815	0.826	0.811
bit	581.772...	8	0.0021	42498627	34	0.8	0.847	0.802
scattering	490.753...	4	0.0091	7290553	22	0.799	0.829	0.788
scattering	559.924...	4	0.0091	7290553	26	0.783	0.791	0.77
bit	172.828...	16	0.0011	42498627	10	0.778	0.804	0.784
scattering	861.383...	16	0.0001	7290553	46	0.776	0.82	0.775
scattering	846.824...	32	0.0021	7290553	46	0.772	0.753	0.748
scattering	559.114...	4	0.0081	7290553	26	0.77	0.824	0.772
scattering	660.705...	16	0.0061	7290553	34	0.767	0.764	0.741
bit	166.389...	16	0.0011	42498627	10	0.763	0.831	0.776
scattering	684.113...	8	0.0021	7290553	34	0.758	0.792	0.753
bit	868.732...	16	0.0001	42498627	46	0.754	0.809	0.759
scattering	492.107...	4	0.0031	7290553	22	0.749	0.745	0.724
resnet50	316.981...	32	0.0130000000...	23514179	26	0.742	0.75	0.735
cnn	200.252...	4	0.0001	1044003	26	0.731	0.75	0.714

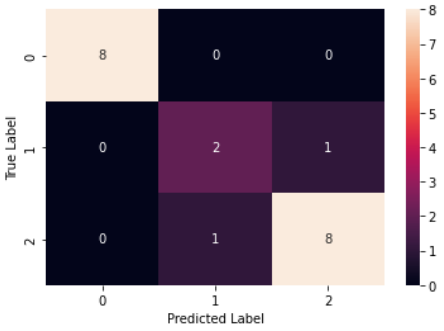
Figure 7: Top hyperparameter search results sorted by macro average recall in descending order.

The architecture that yields the highest macro average recall over the different folds is the BiT with 22 epochs, a mini-batch size of 4 and a learning rate of 0.0031. On average, training time takes 84 seconds per fold. We train the final BiT model using this combination of hyperparameters. The final model was trained on 82 images and evaluated on 20 images. The train and test sets are stratified randomized sets. Figure 8 shows that 4.84GB of GPU memory was used during training. The confusion matrix and classification report is shown in Figure 9.

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 460.80       Driver Version: 460.80       CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  TITAN RTX           Off  | 00000000:65:00.0 Off |                  N/A |
| 41%   45C    P2   133W / 280W |   4624MiB / 24219MiB |     58%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|    0   N/A  N/A    187311      C   ...nvs/oro_health/bin/python     4621MiB |
+-----------------------------------------------------------------------------+
```

Figure 8: Screenshot demonstrating that training the BiT model took 4.84GB of GPU memory.



	precision	recall	f1-score	support
0	1.00	1.00	1.00	8
1	0.75	1.00	0.86	3
2	1.00	0.89	0.94	9
accuracy			0.95	20
macro avg	0.92	0.96	0.93	20
weighted avg	0.96	0.95	0.95	20

Figure 9: (Left) Confusion matrix (Right) Final classification report.

The BiT architecture contains 42M trainable parameters. In Figure 7, we observe the parametric scattering network, which contains 7M trainable parameters, yields a lower macro average recall by only 1.6% than BiT. Thus, to reduce the number of trainable parameters, the parametric scattering network is a good option.

Only one image was misclassified on the test set. The model predicted herpes simplex, but the true label is lichen planus. The model was not confident in the prediction. The probability to be herpes simplex was 65.22%. The misclassified image is shown in Figure 10.

Figure 10: Misclassified Image. The true label is lichen planus. The model predicted herpes simplex.

6 Inference

The state dictionary of the final BiT model was saved in the repository. It takes a total of 7 µs CPU times to create the model and load the state dictionary. A

dummy dataset of acne, lichen planus and herpes simplex images was created using images from the Internet. Using the trained model, the inference was performed on this new dataset. It takes 0.53 µs CPU times per image to do inference. The predictions are shown in Figure 11.

	pred	image_path	pred_class
0	1	random_images/lichen2.jpeg	herpes_simplex
1	0	random_images/acne3.jpeg	acne
2	0	random_images/acne2.jpeg	acne
3	1	random_images/herpes2.jpeg	herpes_simplex
4	2	random_images/lichen3.jpg	lichen_planus
5	1	random_images/herpes.jpeg	herpes_simplex
6	1	random_images/acne.jpeg	herpes_simplex
7	0	random_images/lichen4.jpeg	acne
8	2	random_images/lichen1.jpeg	lichen_planus
9	1	random_images/herpes3.jpeg	herpes_simplex
10	1	random_images/herpes4.webp	herpes_simplex
11	0	random_images/acne4.jpeg	acne
12	2	random_images/lichen.jpeg	lichen_planus
13	1	random_images/herpes1.jpeg	herpes_simplex

Figure 11: Predictions on dummy dataset created from Internet.

7 Limitations

The major limitation of this work has been time. Since time was limited, several design decisions were made to save time. For example, the use of data augmentation has not been explored. The hyperparameter search was not done exhaustively.

8 Possible Improvements

The time constraint limited the scope of the work. That said, this section discusses the possible improvements.

1. The hyperparameter search was quite simple. A more exhaustive search is needed to find the best combination of hyperparameters.

2. The only data augmentation that is applied is random horizontal flip. Since we are in a small data regime, data augmentations can help avoid overfitting. Investigating the use of various data augmentation is needed.

3. Reading the literature to find advanced techniques used for similar tasks is essential. Due to lack of time, only a few papers were read. More research is needed.

4. We should also consider the use of different optimizers and schedulers to vary the learning rate during the training. For example, we should consider the one cycle scheduler that improves convergence during optimization, especially in the small data regime, due to its so-called super convergence policy [11].

5. In Gauthier et al. [4], it is shown that replacing the linear layer on top of the parametric scattering network for a wide residual network yields state-of-the-art results on CIFAR-10. For this work, we should also evaluate this approach.

6. Since the images have different sizes, the images were resized to 400 pixels x 400 pixels. More exploration is needed to find the best way to handle the problem of the different size.

7. All the hyperparameters and settings are hardcoded in the various notebooks. A best practice is to put all of the hyperparameters and settings in a separate file (such as a YAML file).

8. MLflow allows the user to save artifacts such as figures and files. For each run, it would have been interesting to save all the confusion matrices and train loss plots using MLfLow. We can also save the PyTorch model in MLflow.

9 Reproducibility Statement

The results presented in this work are reproducible. The work is available on `https://github.com/sgaut023/skin_disease_classification`. The content of this link contains instructions on how to run the different notebooks to reproduce the results and use the trained model to infer unseen images.

10 References

[1] ALOM, M. Z., TAHA, T. M., YAKOPCIC, C., WESTBERG, S., SIDIKE, P., NASRIN, M. S., HASAN, M., VAN ESSEN, B. C., AWWAL, A. A., AND ASARI, V. K. A state-of-the-art survey on deep learning theory and architectures. *Electronics 8*, 3 (2019), 292.

[2] BYRA, M., STYCZYNSKI, G., SZMIGIELSKI, C., KALINOWSKI, P., MICHAŁOWSKI, Ł., PALUSZKIEWICZ, R., ZIARKIEWICZ-WRÓBLEWSKA, B., ZIENIEWICZ, K., SOBIERAJ, P., AND NOWICKI, A. Transfer learning with deep convolutional neural network for liver steatosis assessment in ultrasound images. *International journal of computer assisted radiology and surgery 13*, 12 (2018), 1895–1903.

[3] CHOUHAN, V., SINGH, S. K., KHAMPARIA, A., GUPTA, D., TIWARI, P., MOREIRA, C., DAMAŠEVIČIUS, R., AND DE ALBUQUERQUE, V. H. C. A novel transfer learning based approach for pneumonia detection in chest x-ray images. *Applied Sciences 10*, 2 (2020), 559.

[4] GAUTHIER, S., THÉRIEN, B., ALSÈNE-RACICOT, L., RISH, I., BELILOVSKY, E., EICKENBERG, M., AND WOLF, G. Parametric scattering networks. *arXiv preprint arXiv:2107.09539* (2021).

[5] KOLESNIKOV, A., BEYER, L., ZHAI, X., PUIGCERVER, J., YUNG, J., GELLY, S., AND HOULSBY, N. Big transfer (bit): General visual representation learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16* (2020), Springer, pp. 491–507.

[6] MALLAT, S. Group invariant scattering. *Communications on Pure and Applied Mathematics 65*, 10 (2012), 1331–1398.

[7] MALLAT, S. Understanding deep convolutional networks. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 374*, 2065 (2016), 20150203.

[8] MENG, D., ZHANG, L., CAO, G., CAO, W., ZHANG, G., AND HU, B. Liver fibrosis classification based on transfer learning and fcnet for ultrasound images. *Ieee Access 5* (2017), 5804–5810.

[9] MINAEE, S., KAFIEH, R., SONKA, M., YAZDANI, S., AND SOUFI, G. J. Deep-covid: Predicting covid-19 from chest x-ray images using deep transfer learning. *Medical image analysis 65* (2020), 101794.

[10] RAVISHANKAR, H., SUDHAKAR, P., VENKATARAMANI, R., THIRUVENKADAM, S., ANNANGI, P., BABU, N., AND VAIDYA, V. Understanding the mechanisms of deep transfer learning for medical images. In *Deep learning and data labeling for medical applications*. Springer, 2016, pp. 188–196.

[11] SMITH, L. N., AND TOPIN, N. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications* (2019), vol. 11006, p. 1100612.

[12] YU, Y., LIN, H., MENG, J., WEI, X., GUO, H., AND ZHAO, Z. Deep transfer learning for modality classification of medical images. *Information 8*, 3 (2017), 91.