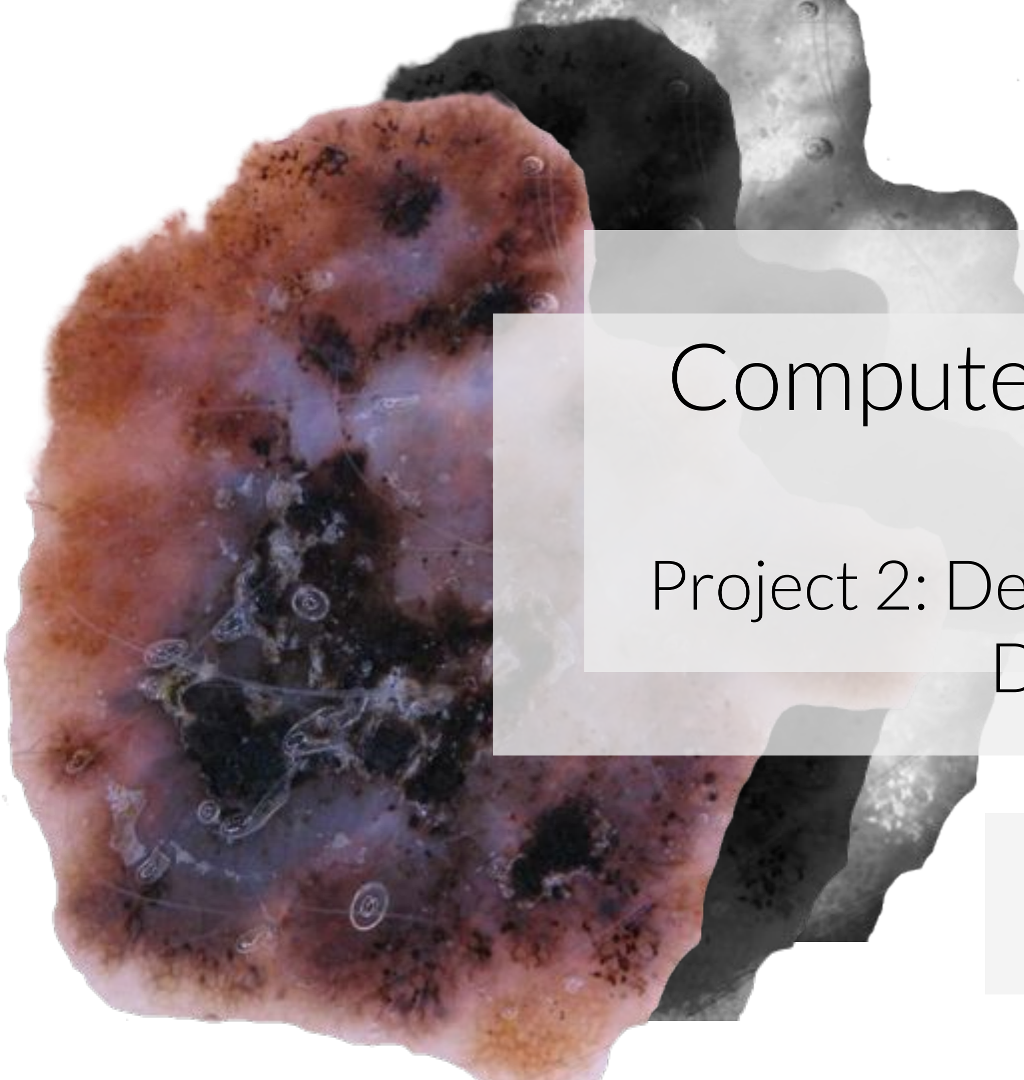


Computed Aided Diagnosis

Project 2: Dermoscopic diagnosis with Deep Learning

Alejandro Cortina Uribe
Cansu Yalcin

Contents

Dataset description

- Challenge 1: Binary

# images	Nevus	Others							
	Total	Total	mel	bcc	bkl	ack	scc	vac	def
Train	7725	7470	2713	1993	1574	520	376	151	143
Val	1931	1865	678	498	393	130	94	37	35

Balanced dataset

Different image sizes

- Challenge 2: Three class

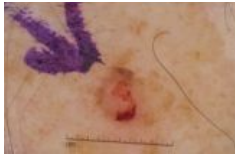
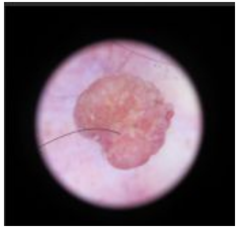
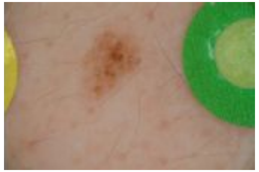
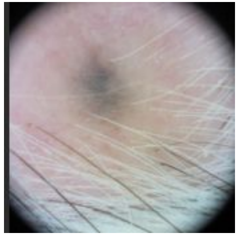
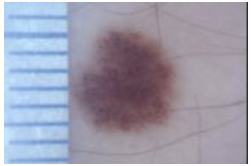
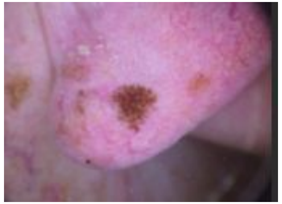
# images	bcc	mel	scc
Train	1993	2713	376
Val	498	678	94

Unbalanced dataset

Different image sizes

Image characteristics:

- Artifacts:
  - Dark and light hairs
  - Marker annotations
  - Rulers
  - Stickers
- Black circular vignette (border)
- Variate non-standard lesion framing.

Preprocessing: FOV Removal and Resizing

FOV Removal algorithm[1]:

1. Obtain the histogram of image diagonals
2. Obtain histograms crossing points with a selected threshold
3. Select the most restrictive cropping coordinates

Problems:

- Non-circular vignette (single horizontal or vertical black border)
- Off-center nearly-black lesion

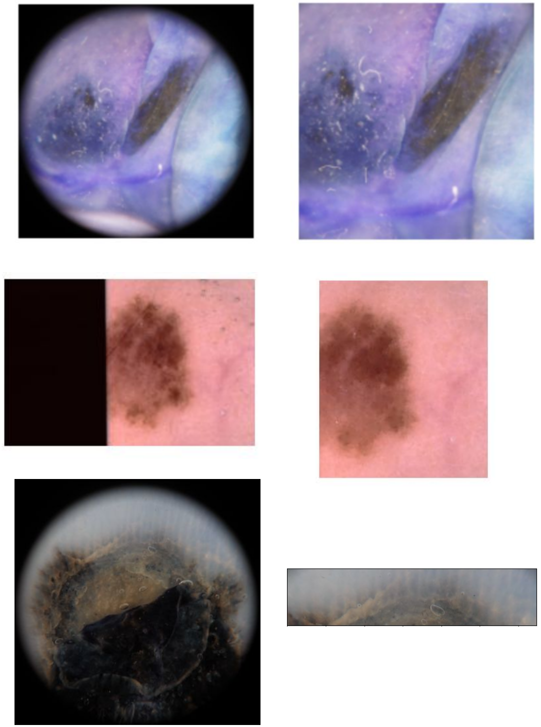
Solutions:

- Remove non-circular vignette
- Obtain very restrictive crop

Resize all the images training / validation / test to 224x224.

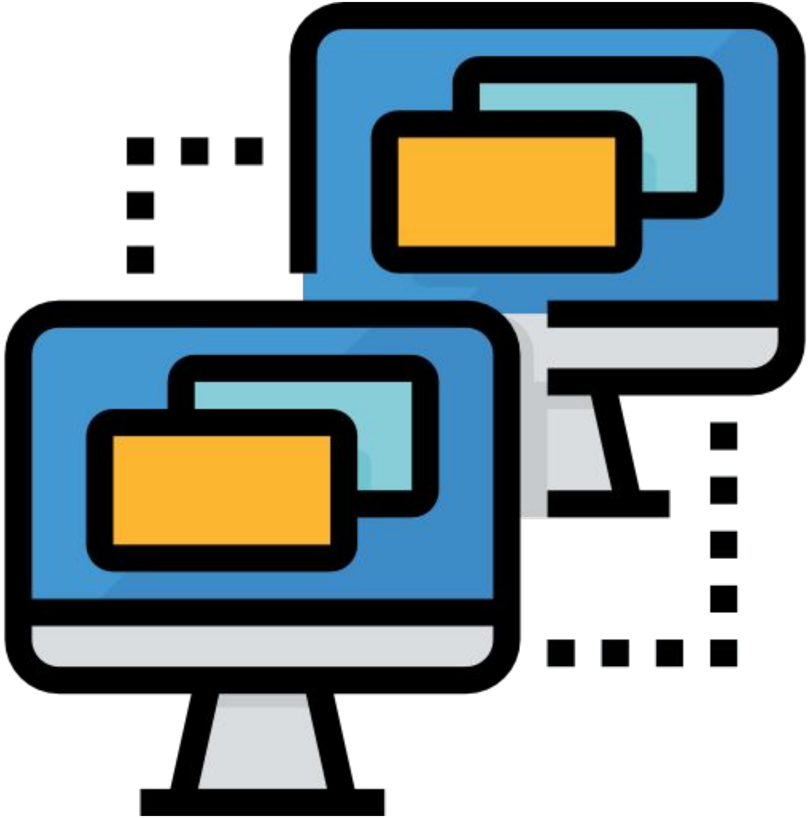
- Faster training times allowing to have more experiments.

Original Cropped



Device Settings

- Experiments were carried out using the server from UdG and Google Colab.
    - UdG server: mariecurie.udg.edu
      Containing 2 GPU'S with 11 GB
    - Google Colaboratory
      Tesla K80 GPU with 12 GB
- Developed with Pytorch



Transfer Learning experiments

As vanilla experiments, we ran some transfer learning training:

Type	Architecture	Trainable params	Accuracy		Notes
			Train	Val	
Fine-tuning	MobileNetV2	2,226,434	0.9338	0.8609	
	Densenet121	6,955,906	**0.9951**	**0.8814**	
	Inception-Resnet-v2	54,309,538	0.9297	0.8762	
	Resnet18	11,690,538	0.9773	0.8759	
Feature extraction	Resnet18	4,098	0.7993	0.8051	
Fine-tuning	Resnet50	25,561,130	0.954	0.8791	
	Resnet50	17,017,834	0.9218	0.8519	Freezing first 7 layers
	Resnet50	24,116,202	0.9466	0.8746	Freezing first 6 layers
	Resnet50	24,116,202	0.8058	0.7964	SGD, LR=0.0001
	Resnet50	24,116,202	0.8646	0.8322	SGD, LR=0.001

Challenge 1

Binary classification

Fine-tuning architectures

Binary

Focusing on widely used architectures for our problem, we used:

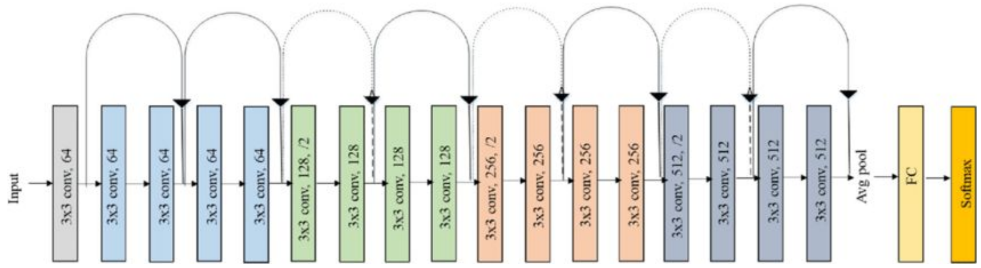
Resnet18 [2]

Densenet121

Regnet

ViT

Swin Tiny



Trainable parameters	Key elements
11,690,538	● Residual connections ● 18 convolutional layers

Fine-tuning architectures

Focusing on widely used architectures for our problem, we used:

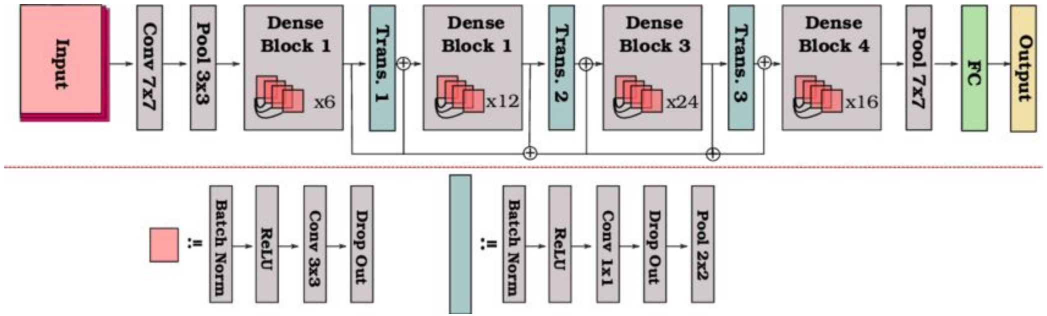
Resnet18

Densenet121 [3]

Regnet

ViT

Swin Tiny



Trainable parameters	Key elements
6,955,906	• 4 Dense blocks • L(L+1)/2 direct connections • Substantial reduction of params

Fine-tuning architectures

Focusing on widely used architectures for our problem, we used:

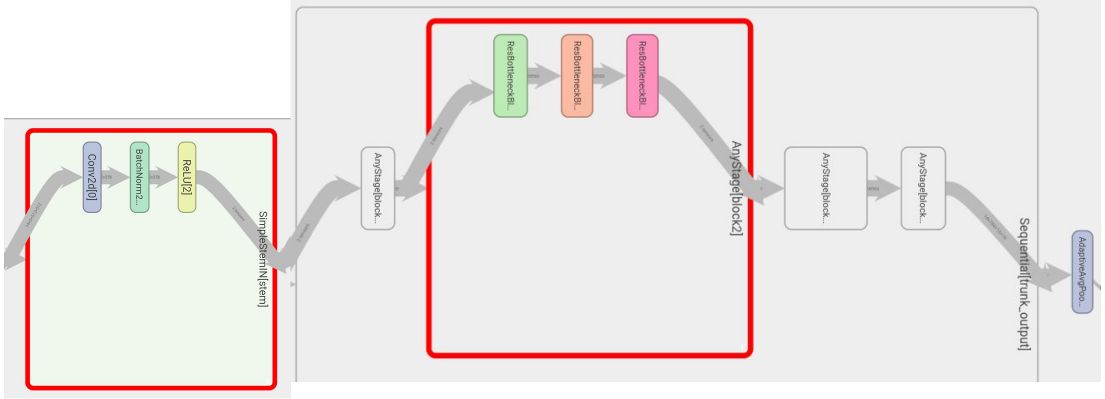
Resnet18

Densenet121

Regnet [4]

ViT

Swin Tiny



Trainable parameters	Key elements
5,649,082	• Design space provides simple and fast networks, with parameters (d, g, w_m, w_a, w_0): • Using RegnetY_800MF

Fine-tuning architectures

Focusing on widely used architectures for our problem, we used:

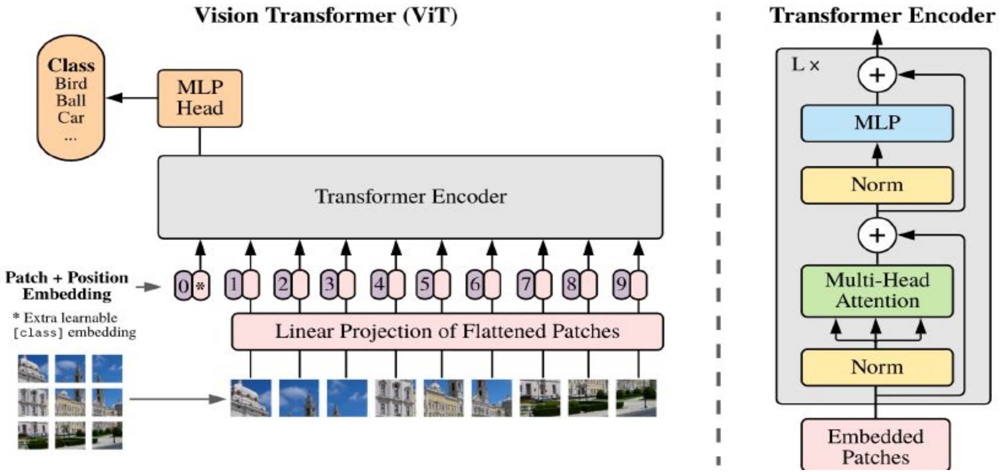
Resnet18

Densenet121

Regnet

ViT [5]

Swin Tiny



Trainable parameters	Key elements
85,800,194	• Layer Norm • Multi-head Attention Network (MSP) • Multi-Layer Perceptrons (MLP)

Fine-tuning architectures

Focusing on widely used architectures for our problem, we used:

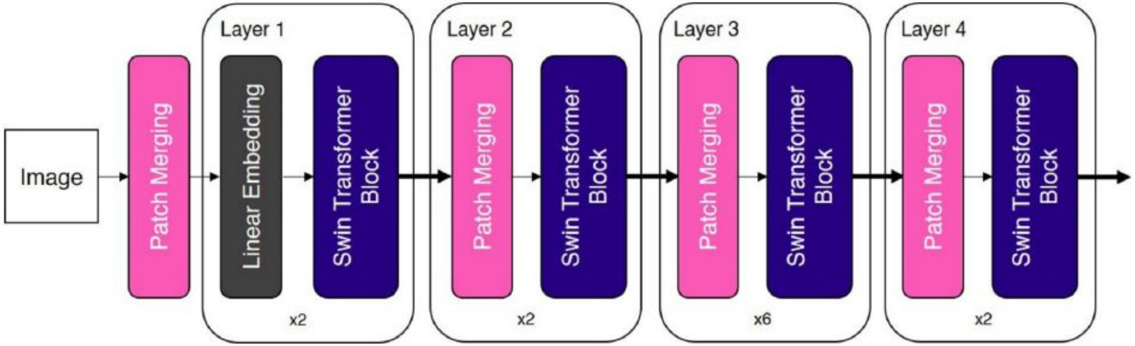
Resnet18

Densenet121

Regnet

ViT

Swin Tiny [6]

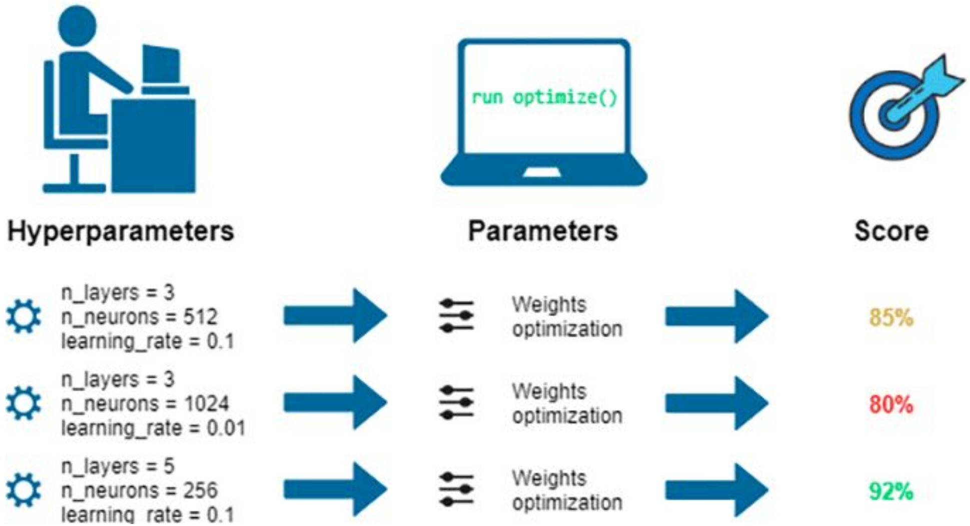


Trainable parameters	Key elements
27,584,108	• Hierarchical feature maps • Shifted window attention

Hyperparameter tuning

The hyperparameter tuning is a necessary step in order to get optimized performance from the models. It includes,

- Data augmentations
- Optimizer
- Scheduler
- Early stopping



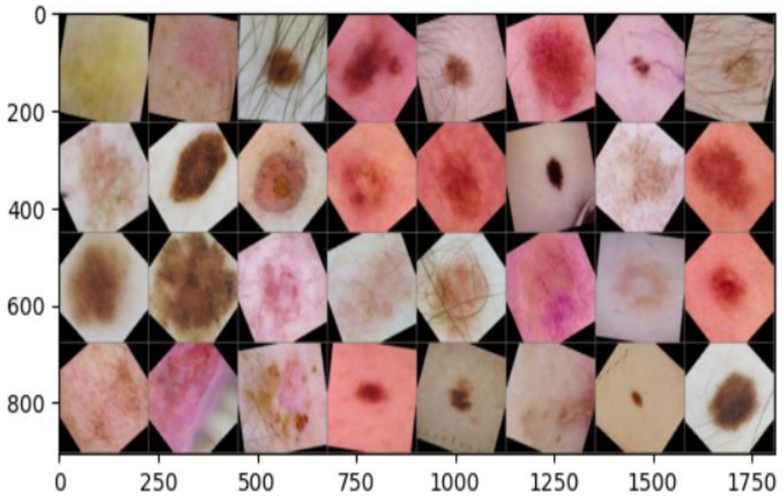
Hyperparameter tuning - Data augmentations

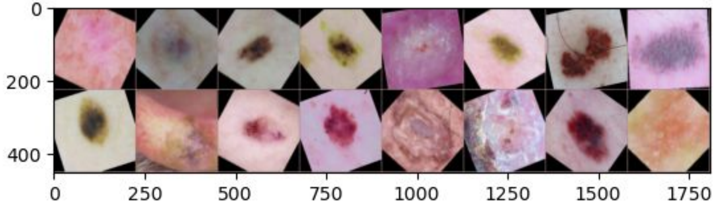
Data augmentations are used to artificially increase the training set by creating modified copies of a dataset using existing data.

They are applied to training set using Pytorch's `torchvision.transforms` module.

- RandomHorizontalFlip
- RandomVerticalFlip
- RandomRotation
- RandomEqualize
- ColorJitter
- GaussianBlur
- ToTensor
- Normalize*

*We have experimented with both the mean and standard deviation of our dataset and ImageNet's.



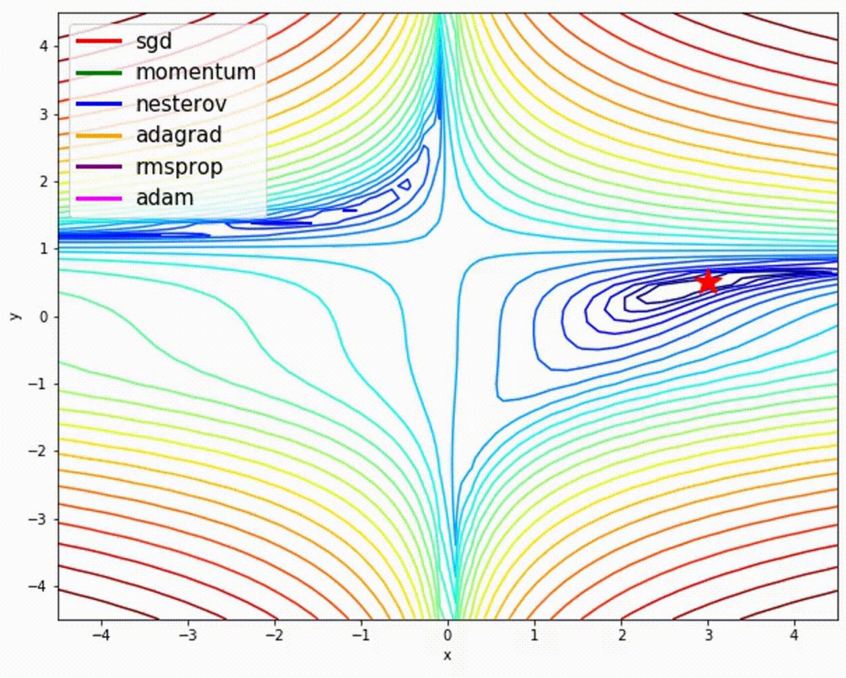


Optimizers are used to adjust the parameters for a model in order to minimize/maximize the loss function.

- ADAM (Adaptive moment estimation)
- SGD (Stochastic gradient descent)

Learning schedulers are used to adjusts the learning rate between epochs or iterations as the training progresses.

- StepLR
- ReduceLROnPlateau



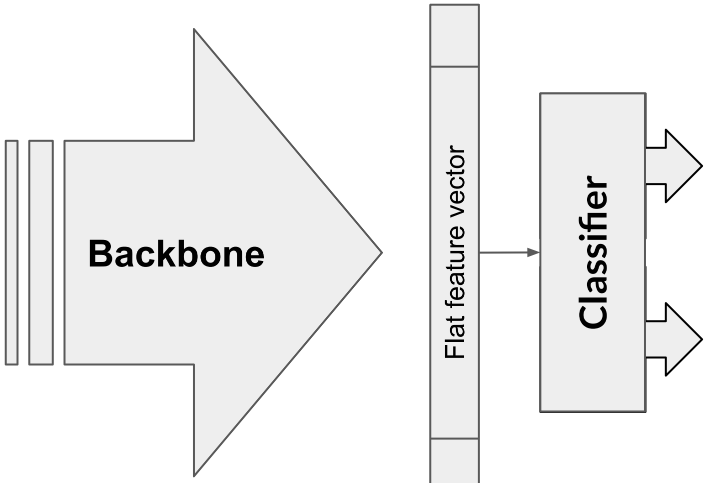
Results

- Best experiments per model

Selected to be used in the ensemble model

Model	Accuracy	
	Train	Validation
Swin_t	0.9582	0.8938
Regnet	0.9785	0.8930
Densenet121	0.9791	0.8898
Swin_t (MLP)	0.9026	0.8825
Resnet50	0.954	0.8791
ViT	0.9371	0.8743
MobileNetV2	0.9338	0.8609
Resnet18	0.9175	0.8567

- Best model as feature extractor

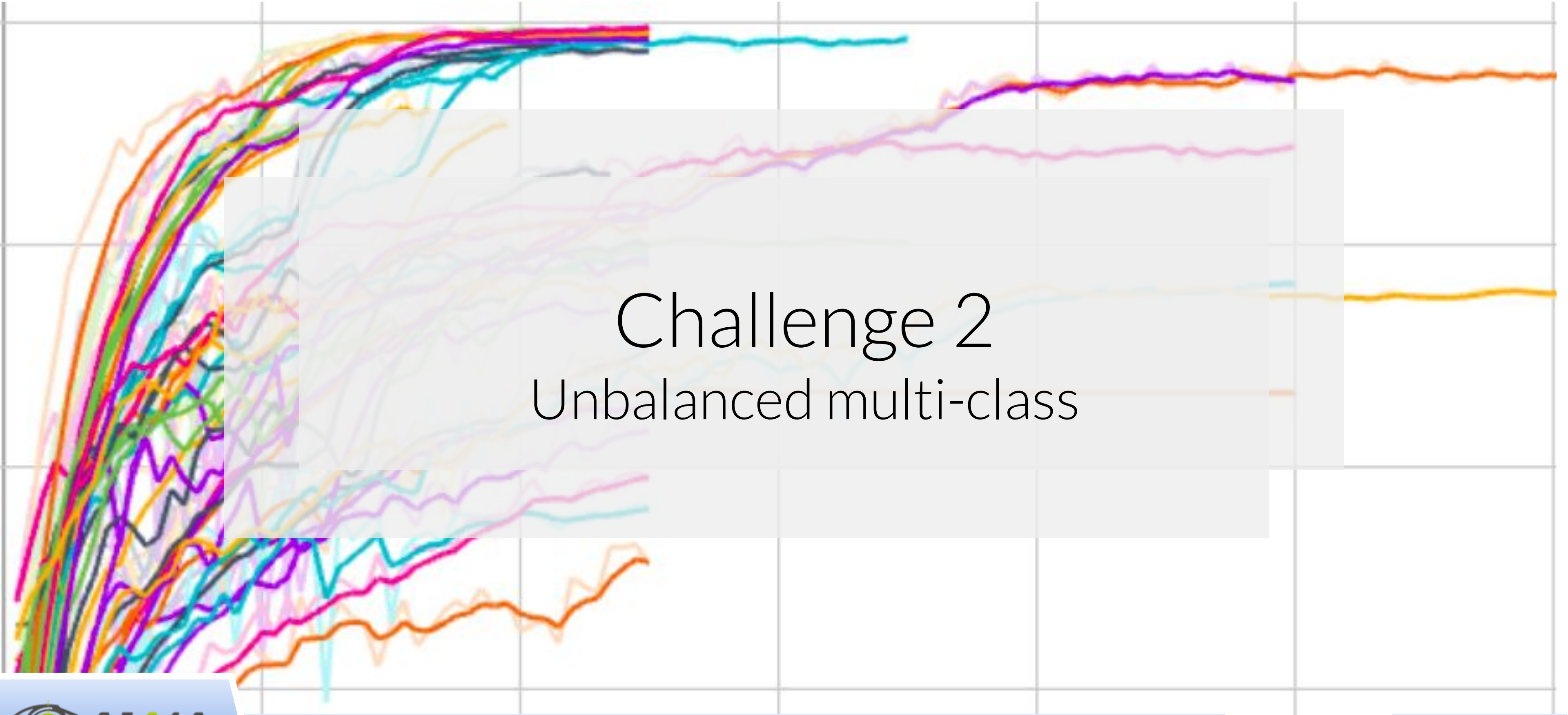


Classifier	Accuracy	
	Train	Validation
Backbone and FC layer	0.9079	0.8806
SVM avg_pool1d → 258 features RBF kernel C=1, Gamma=0.1	0.9229	**0.8843**
MLP ReLU(Linear(768, 512)) ReLU(Linear(512, 256)) Linear(256, 2)	0.9026	0.8825

Ensemble predictions

- We have performed an ensemble of the best models:
  - Majority voting
  - Soft voting

Type	Model's assembly	Mean validation accuracy
MV	DN(exp6)+SW(exp15-4)+RGN(ex1)	0.91119
MV	SW(exp15-4)+SW(exp15)+SW(exp15-4-3)	0.9
SV	DN(exp6)+SW(exp15-4)+RGN(ex1)	**0.911718**
MV	DN(exp6)+SW(exp15-4)+RGN(ex1)+SW(exp15)+SW(exp15-4-3)	0.910937
SV	DN(exp6)+SW(exp15-4)+RGN(ex1)+SW(exp15)+SW(exp15-4-3)	0.9117187



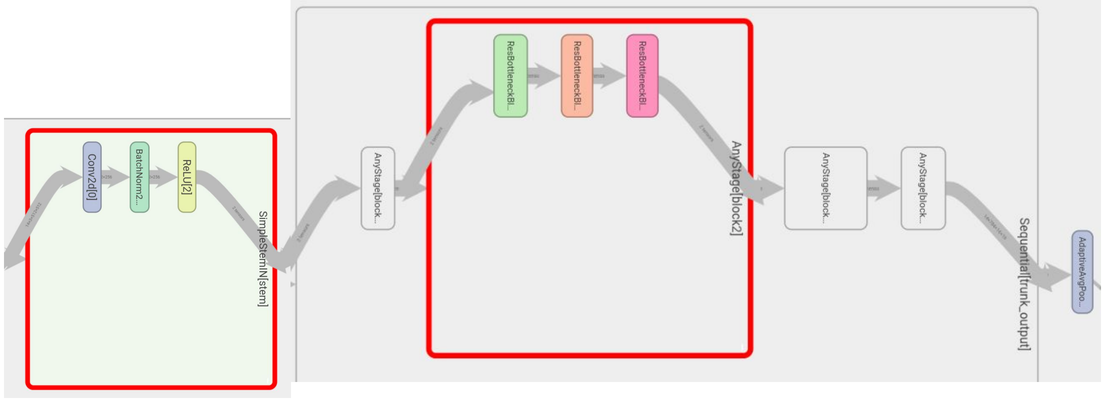
Challenge 2

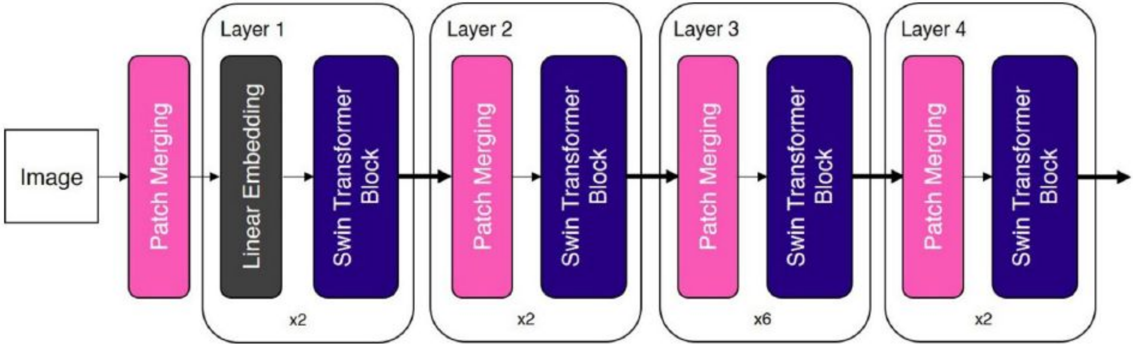
Unbalanced multi-class

Fine Tuning Architectures

The architectures we used for the multi-class problem:

- Regnet
- Swin Tiny





Hyperparameter Tuning

- To address the class imbalance we tested two losses and a data sampler.

The **loss functions** we tested were:

- Cross Entropy, with weights (as total_samples/class_count)
- Focal Loss [7].

We tested the **WeightedRandomSampler**, so for each batch, the dataloader samples with replacement according to sample weights.

Learning schedulers are used to adjusts the learning rate between epochs or iterations as the training progresses.

- MultiStepLR
- ReduceLROnPlateau

- Best experiments per model

Selected to be used in the ensemble model

Model	Accuracy		Kappa	
	Train	Validation	Train	Validation
Swin_T	**0.9988**	**0.9551**	**0.9979**	**0.919**
Swin_T (Focal loss + modified head)	0.988	0.9043	0.9873	0.9117
Swin_T (with multistep scheduler)	0.9925	0.909	0.9943	0.9008
Regnet	0.9989	0.9086	0.9926	0.9076

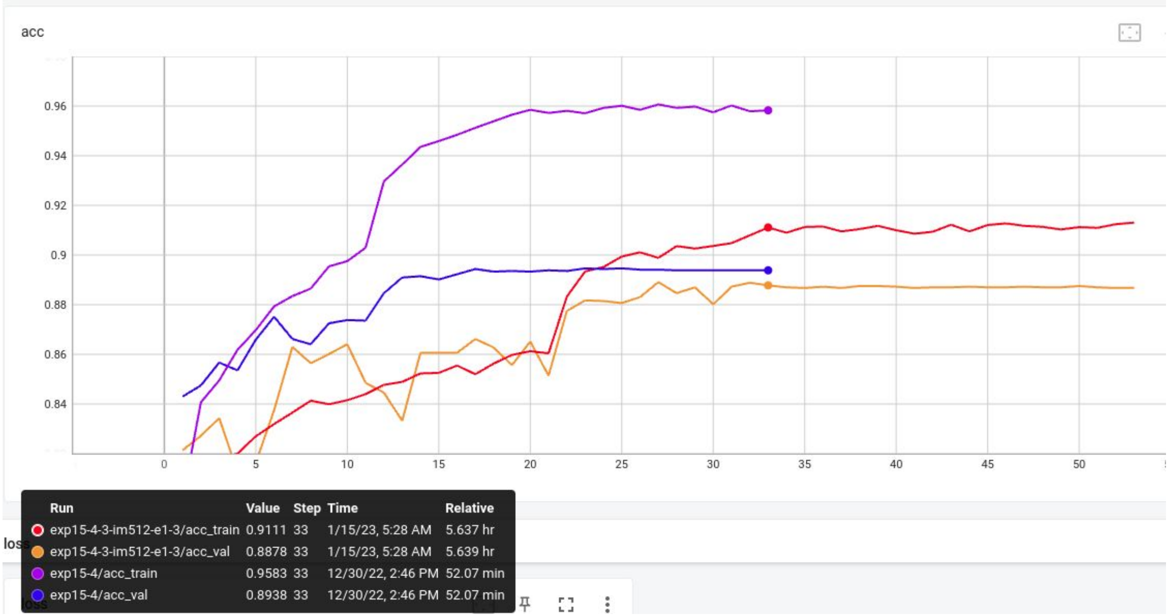
Ensemble predictions

- We have performed an ensemble of the best models:
  - Majority voting
  - Soft voting

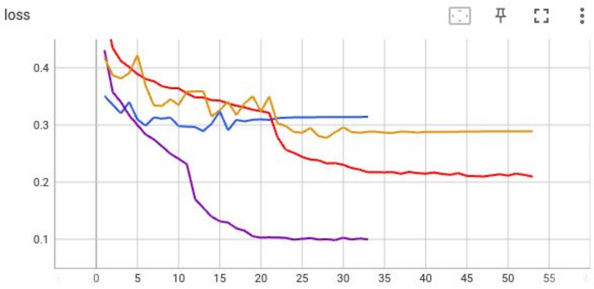
Type	Model's assembly	Mean validation kappa
MV	SW(exp1) + SW(exp9) + SW(exp13)	**0.922**
MV	SW(exp1) + SW(exp9) + SW(exp13)+RGN(exp4)	0.913
SV	SW(exp1) + SW(exp9) + SW(exp13)	0.902
SV	SW(exp1) + SW(exp9) + SW(exp13)+RGN(exp4)	0.89
MV	SW(exp1) + SW(exp6) + SW(exp13)	0.87

Final experiments

- To test the image size as a factor of increased performance, we ran final experiments with images of size 512x512

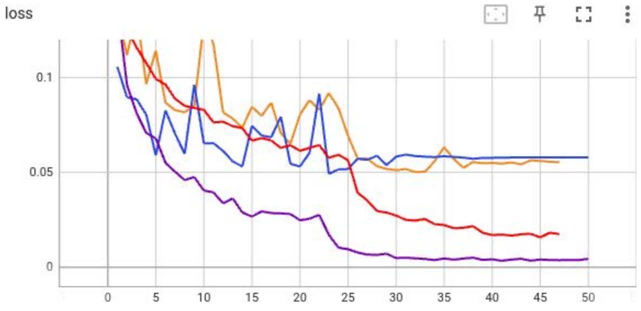


- ❖ Less overfitted model compared to best 224 model
- ❖ Didn’t reach best 224 model performance



- To test the image size as a factor of increased performance, we ran final experiments with images of size 512x512

- ❖ Less overfitted model compared to best 224 model
- ❖ Didn’t reach best 224 model performance



Conclusions

- We experimented we different pre-trained deep learning models and obtaining their best training parameters.
- We reduced drastically the image size in order to perform proper experiments for the hyperparameters' tuning.
- It is particularly challenging to train complex models, to reach higher performance while monitoring the model's overfitting.
- For our validation set, we obtained an accuracy score of 0.9117 for the binary challenge, and kappa score of 0.922 for the multi-class challenge by using ensemble methods.
- Thus concluding that the selected models were able to well characterize the two lesion and multiple lesion classes.

References

[1] https://github.com/lcambero/skin_lesion_segmentation

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[3] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. CoRR, abs/1608.06993, 2016

[4] Iija Radosavovic, Raj Prateek Kosaraju, Ross B. Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. CoRR, abs/2003.13678, 2020.

[5] https://github.com/google-research/vision_transformer

[6]Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., & Guo, B. (2021). Swin Transformer: Hierarchical vision transformer using shifted windows. 2021 IEEE/CVF International Conference on Computer Vision (ICCV). https://doi.org/10.1109/iccv48922.2021.00986

[7] Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal Loss for Dense Object Detection, 130(4), 485–491. https://doi.org/10.1016/j.ajodo.2005.02.022

This project was developed in the beautiful city of Girona, Catalonia

Not in Spain jeje