Master in Computer Vision Barcelona

M3 – Machine Learning for Computer Vision

Project: Deep learning classification - **Session 4**

Group 7: Guillem Capellera, Johnny Nuñez and Anna Oliveras

Tasks

Understanding layer manipulation

0. Fine tune an existing architecture
1. Set a new model from an existing architecture
2. Apply the model to a small set of data (no more than 400)

Deal with dataset loading

3. Introduce and evaluate the usage of data augmentation

Hyperparameter optimization

4. Introduce and evaluate the usage of any suitable methodology to improve learning curve (dropout layer, batch norm, …)
5. Apply random search / optuna on per model hyperparameters

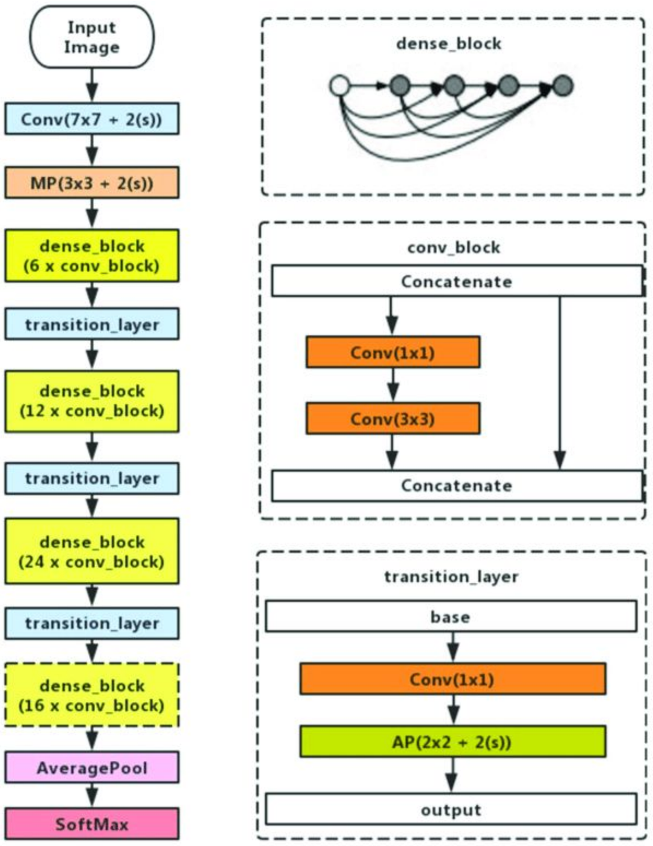
Datasets

- We have 8 classes: coast, forest, highway, inside city, mountain, open country, street, tall buildings
- Big dataset : MIT_split → total of 2288 images
- Small dataset: MIT_small:train_1 → Train with only 50 images for each class !

DenseNet121 [1]

Size (MB)	33
Top 1-Accuracy	75%
Top 5-Accuracy	92.3%
Parameters	8.1M
Depth	242

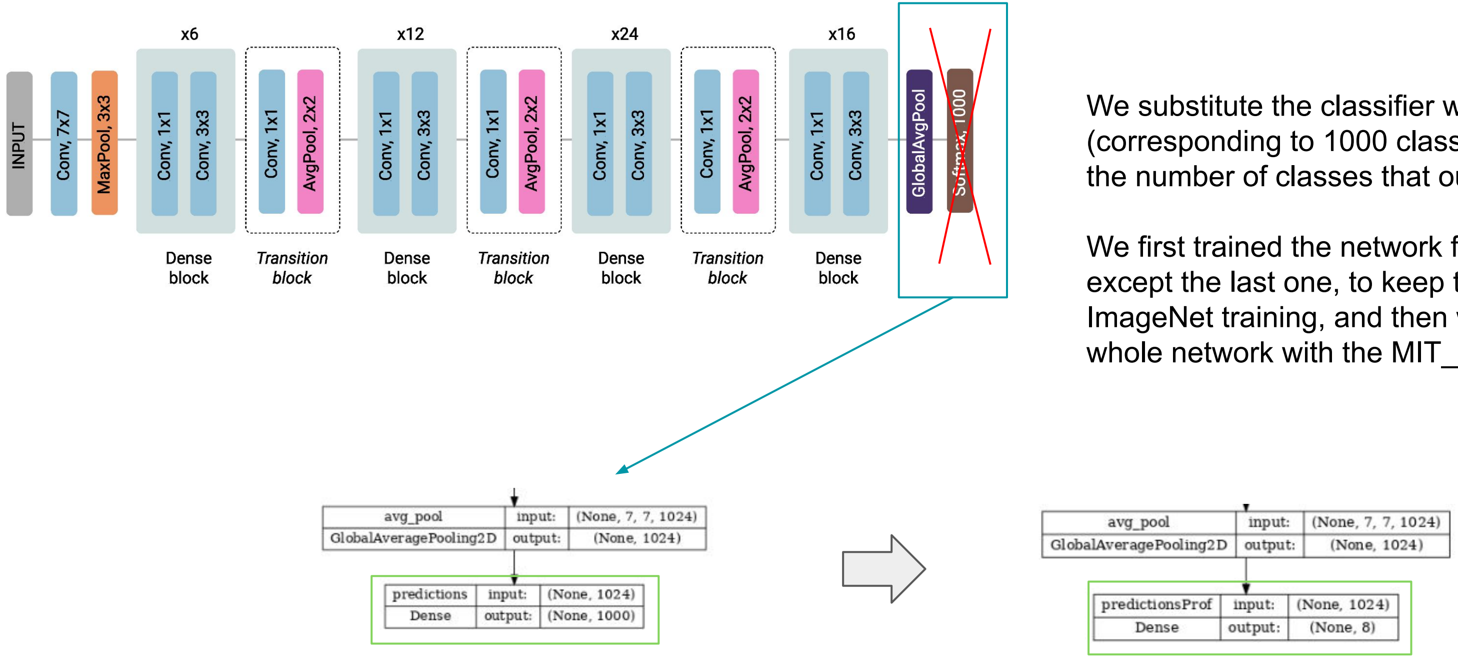
*Performance on ImageNet[2] dataset



[1] Huang, G., Liu, Z., Weinberger, K. Q., & van der Maaten, L. (2017). Densely connected convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 4700-4708). https://arxiv.org/abs/1608.06993

[2] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition* (pp. 248–255).

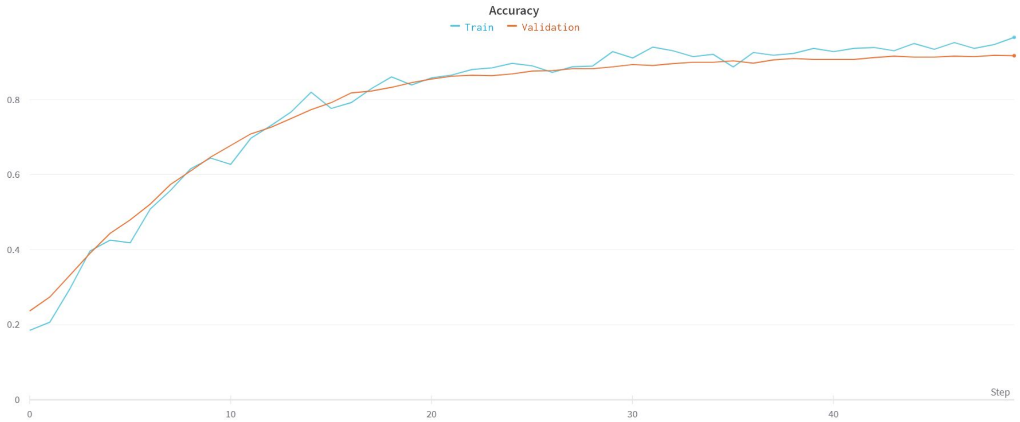
Task 0: Fine tune an existing architecture

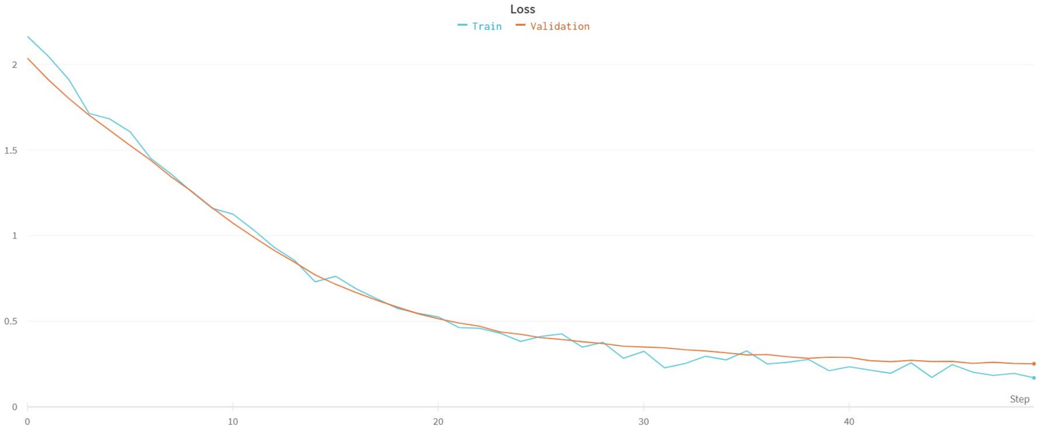


We substitute the classifier with output 1000 (corresponding to 1000 classes of Imagenet) to **8**, the number of classes that our dataset has.

We first trained the network freezing all the layers except the last one, to keep the weights from the ImageNet training, and then we re-trained the whole network with the MIT_split dataset.

Task 0: Fine tune an existing architecture





We reach the following **accuracy**:

- Train : 0.966
- **Validation: 0.917**

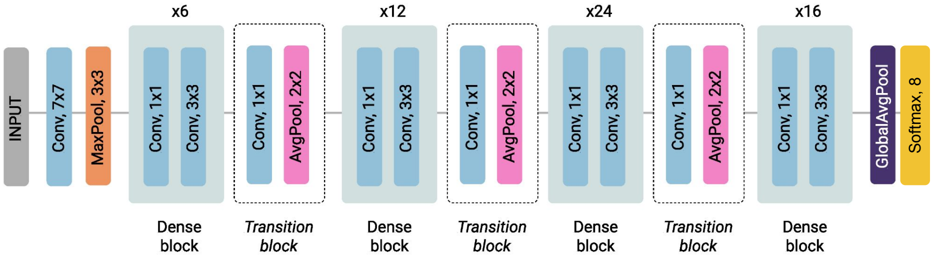
Using the whole MIT dataset, we see that the curves are well defined, the losses are well minimized and we don't have overfitting.

Note that the Train curves are less stable, which may be caused by the optimizer and learning rate or the use of small batches on the training data.

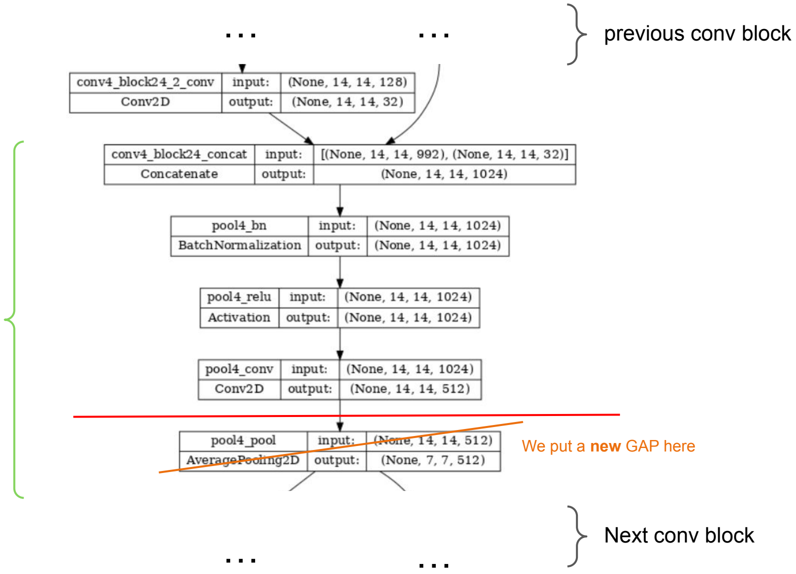
Task 1: Set a new model from an existing architecture

We will try to remove some blocks to reduce the number of parameters, while maintaining a similar accuracy.

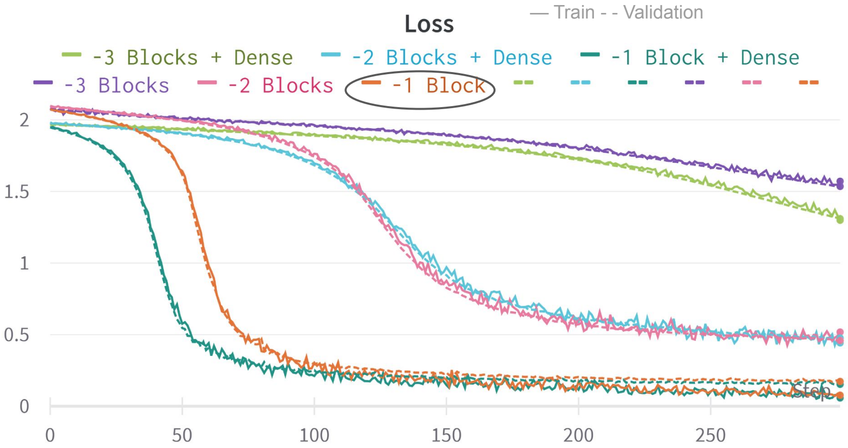
To do so, we will cut the model in the transition block before the dense block we want to remove.

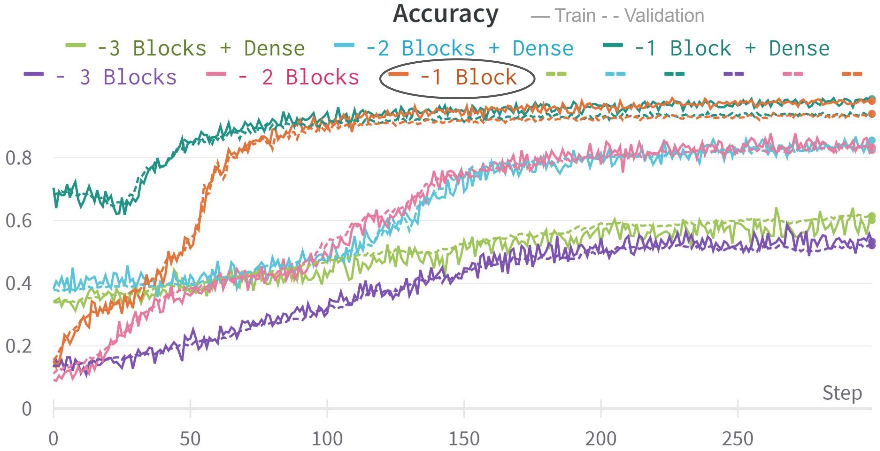


Model	Epochs	Num parameters	Validation accuracy
Original	50	7M	0.9318
Original	300	7M	0.9542
Removing 1 DB (x16)	**300**	**5M**	**0.941**
Removing 2 DB (x24,x16)	300	1.5M	0.825
Removing 3 DB (x24,x16,x12)	300	380.000	0.52
Removing 1 DB + adding dense [1024]	300	5M	0.9393
Removing 2 DB + adding dense [1024]	300	1.5M	0.832
Removing 3 DB + adding dense [1024]	300	520.000	0.601



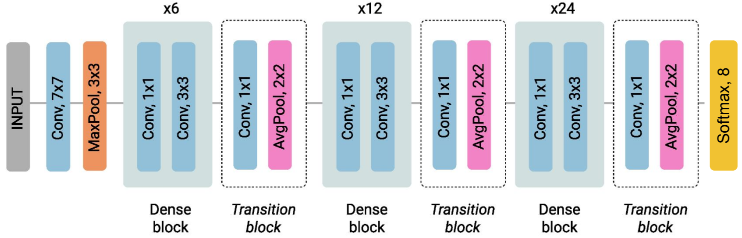
Task 1: Set a new model from an existing architecture





We choose the orange one as our new model, since we can maintain a **similar accuracy** while reducing the number of parameters (**29% less parameters** than the original model).

This may be because the model needed to be more complex to classify a lot of different classes like ImageNet, but in our case, as we only have 8 classes, the New model is enough.

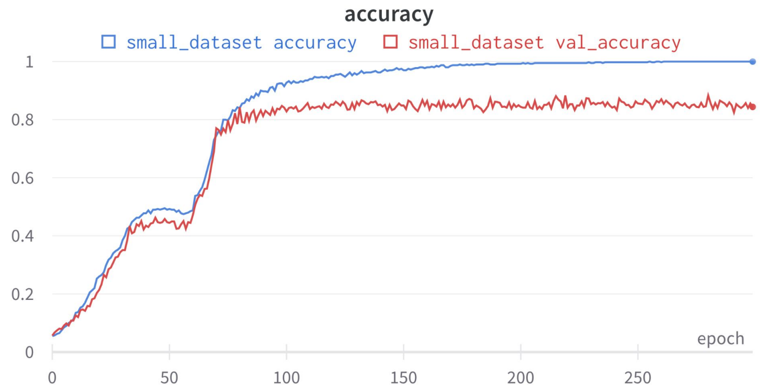


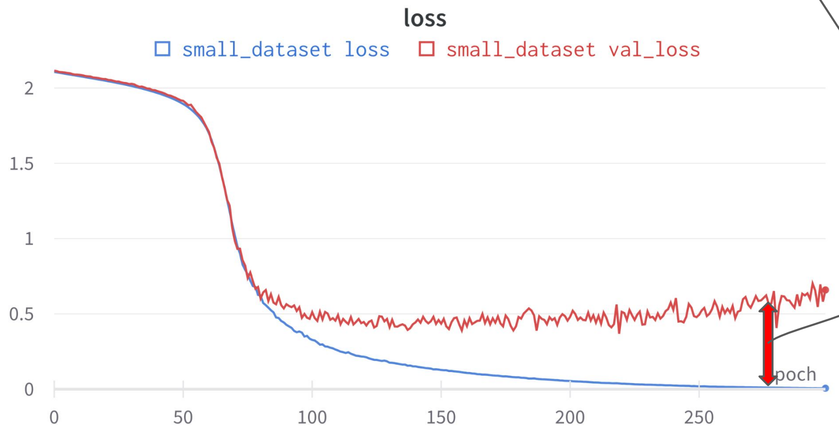
Model	Num Parameters	Validation accuracy
New Model	5M	0.941

Task 2: Train the model with smaller dataset

Notice that when we are training the model using the small dataset, **overfitting** occurs. It was expected since with less data, the model has less generalization power.

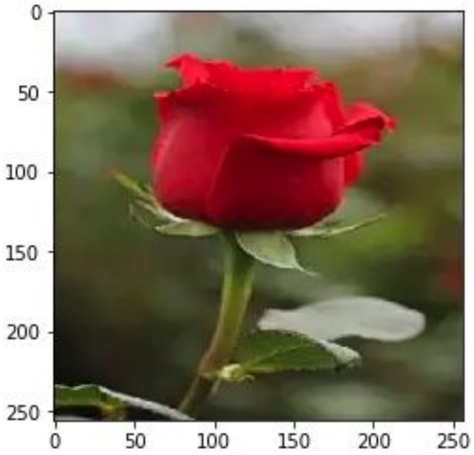
A possible solution is to introduce data augmentation to artificially increase our dataset.



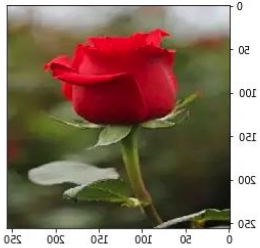


Dataset	Validation accuracy
MIT	0.941
MIT_small_1	0.845

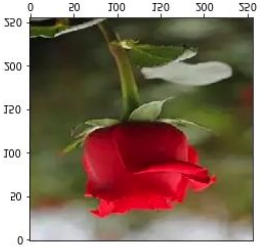
Task 3: Data augmentation



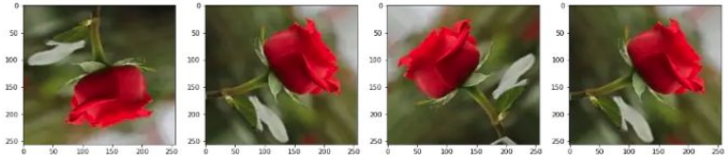
Horizontal flip



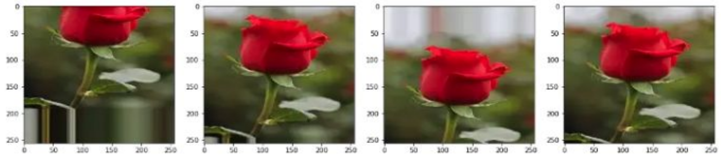
Vertical Flip



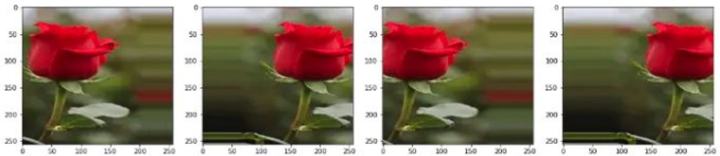
Rotation



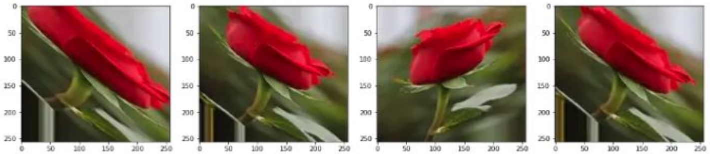
Height Shift



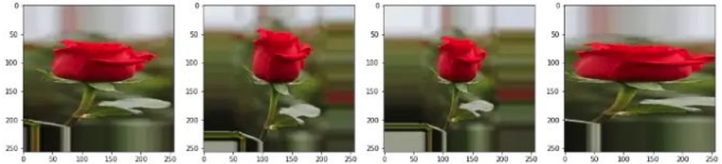
Width Shift



Shear Intensity



Zoom Range



https://medium.com/analytics-vidhya/understanding-image-augmentation-using-keras-tensorflow-a6341669d9ca

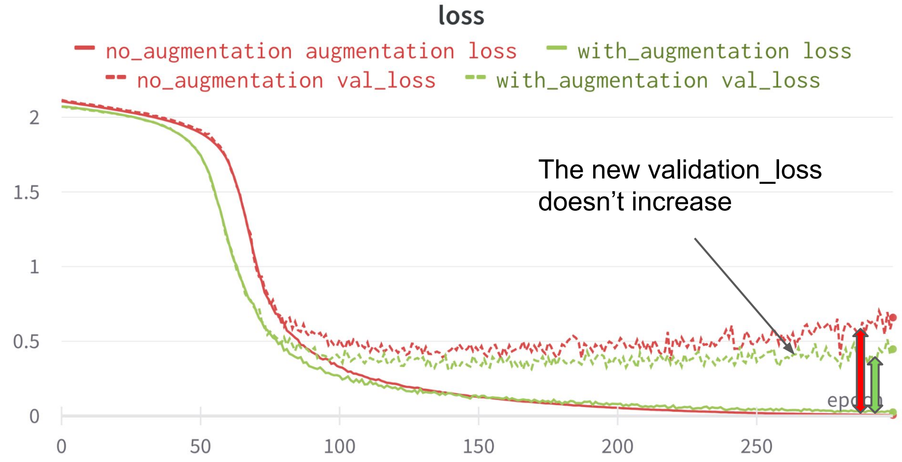
Task 3: Data augmentation

We have used a grid search where the following values have been set:

Data augmentation	Values
Rotation	0º , 20º
Height Shift	0%, 20%
Width Shift	0%, 20%
Shear Intensity	0%, 20%
Zoom Range	0%, 20%
Horizontal Flip	False, True
Vertical Flip	False, True

All data augmentation that varies significantly the vertical position (like Vertical Flip) reduces the accuracy of the model.

Best combination



Best combination is using **Horizontal Flip = True** and a **Zoom Range = 20%**. With this two data augmentation techniques we increase the accuracy and decrease the gap between loss curves.

Dataset	Validation accuracy
MIT	0.941
MIT_small_1	0.845
MIT_small_1 data augmentation	**0.895**

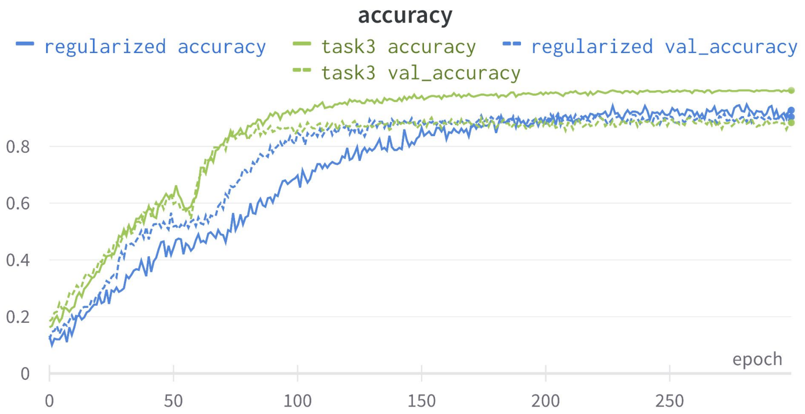
Task 4: Improve learning curve

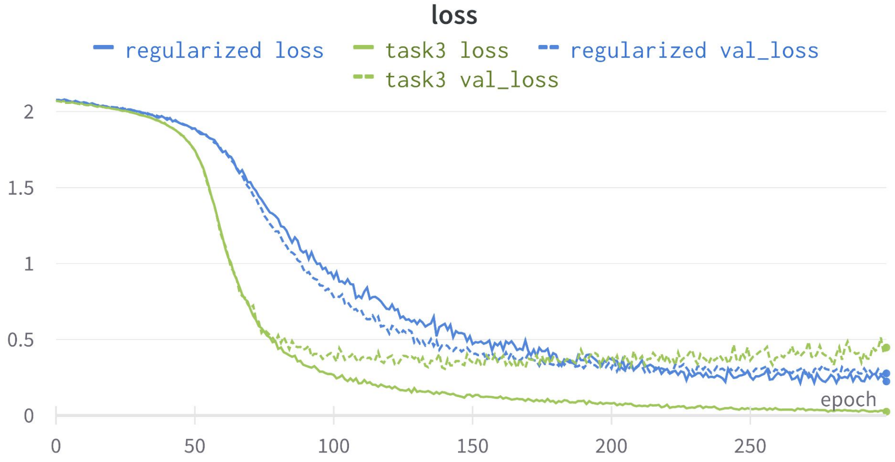
In this task, we add some techniques in order to obtain a proper learning curve:

- **Early Stopper**: regularization technique used to prevent overfitting in machine learning by terminating the training process when the validation loss stops improving.
- **Reduce lr:** callback function that adjusts the learning rate of the optimizer when the validation loss has stopped improving.
- **BatchNormalization:** applies a transformation that maintains the mean output close to 0 and the output standard deviation close to 1 (in fact the initial model includes batch normalization at every block).
- **Dropout:** randomly sets input units to 0 with a frequency of rate at each step during training time, which prevents overfitting. We've performed a grid search with different % values of the dropout. → **best value 50%**

Task 4: Improve learning curve

Dropout helps to reduce overfitting by preventing any one neuron from having too much influence over the output, as well as encouraging the network to learn multiple independent representations of the data





Dataset	Validation accuracy
MIT	0.941
Task2 (small dataset)	0.845
Task3 (data augmentation)	0.881
Task 4	**0.915**

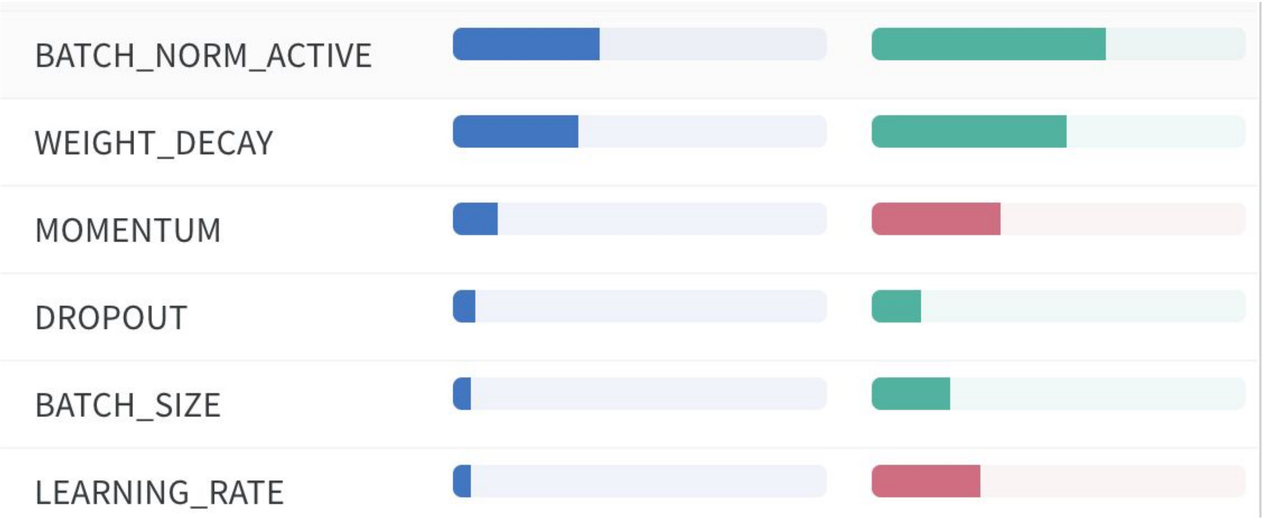
Task 5: Hyperparameters optimization

We have tried 30 trials of hyperparameter tuning, some examples and best model are shown in the table

Model	Optimizer	Weight Decay	Momentum	Dropout	Learning Rate	Batch Size	Batch Normalization	Best Epoch	Accuracy
Model 1	Nadam	0.01	/	0.5	0.00001	10	/	23	0.9383
Model 2	Adamax	0.3	/	0.5	0.0001	10	True	42	0.9444
Model 3	**Adamax**	**0.3**	**/**	**0.5**	**0.0001**	**10**	**True**	**35**	**0.9518**
Model 4	Adagrad	0.001	/	0.5	0.0001	128	True	180	0.88
Model 5	Adamax	0.00001	/	0.5	0.00001	64	False	173	0.917
Model 6	SGD	0.001	0.8	0.5	0.00001	64	True	295	0.79
Model 7	Adadelta	0.001	0.5	0.5	0.0001	10	False	0	0.21
Model 8	Adam	0.1	/	0.5	0.00001	128	True	135	0.92

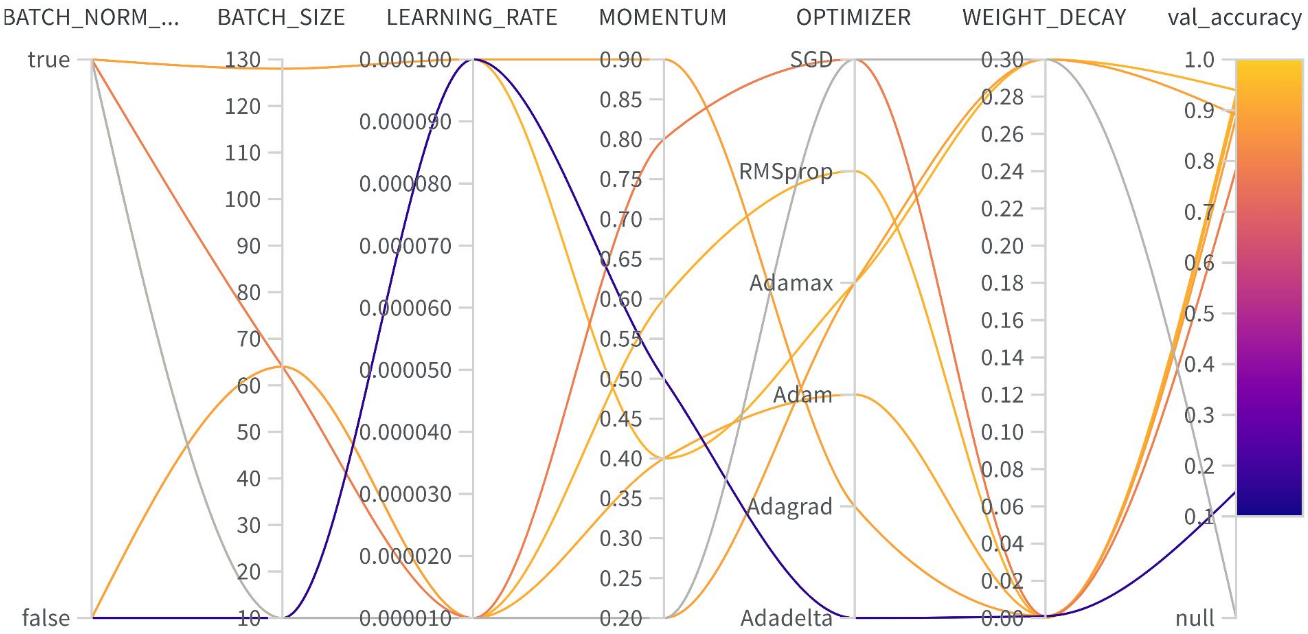
Task 5: Hyperparameters optimization

- **Adam** and its **variations** give the best performance/epoch results.
- **Dropout** and batch normalization increment the performance.



Task 5: Hyperparameters optimization

Some Examples



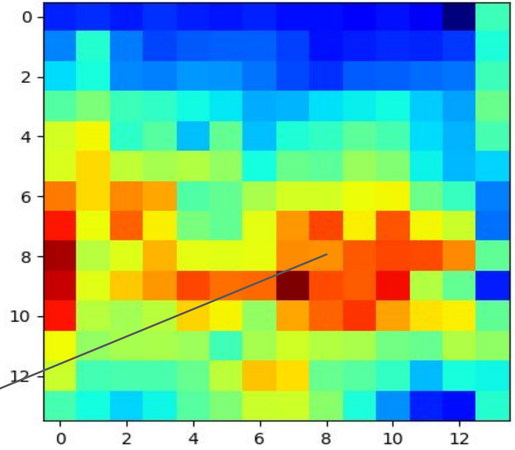
Good Optimizer and Regularization **is all you need**

Extra task → Visualization (Conv4 Block 5: DenseNet 121(BM))

GradCam

Coast

Feature Map



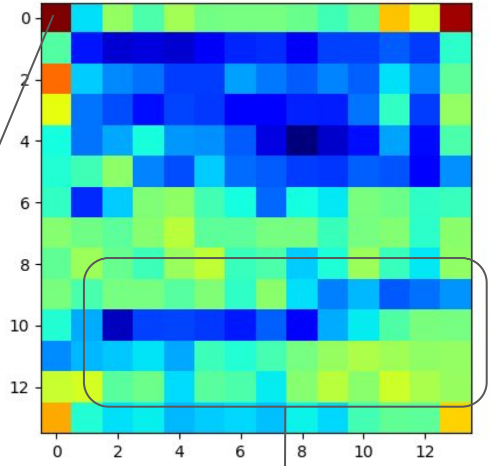
looking at shapes, curves … Big patterns

Extra task → Visualization (Last Conv: DenseNet 121(Best model))

GradCam

Coast

Feature Map



illumination may influence and lead to misclassifications.

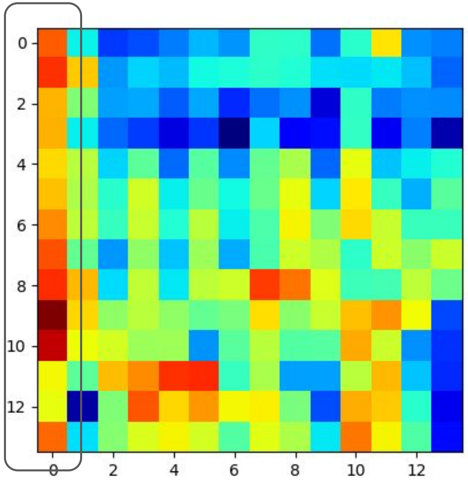
looking at more complex patterns → change between sea and sand

Extra task → Visualization (Conv4 Block 5: DenseNet 121(BM))

GradCam

Inside City

Feature Map



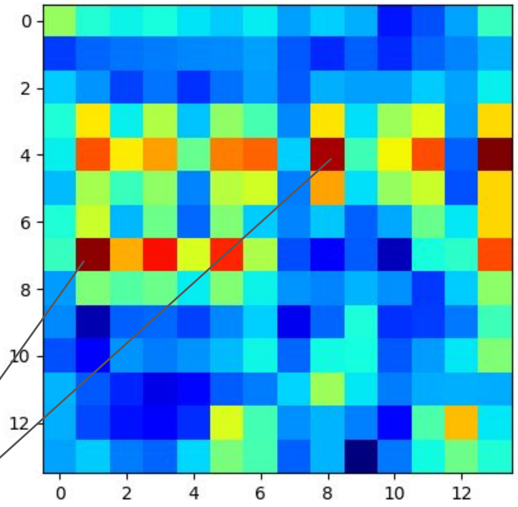
looking at shapes, lines …

Extra task → Visualization (Last Conv: DenseNet 121(Best model))

GradCam

Inside City

Feature Map

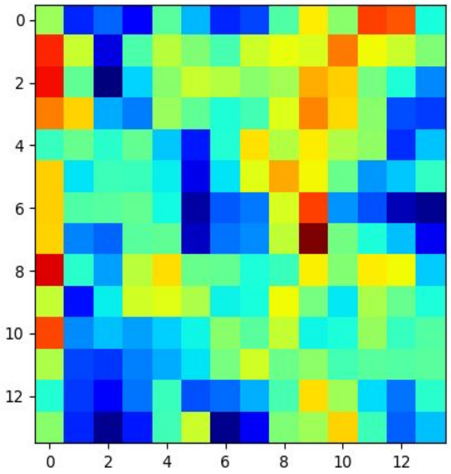


Complex patterns like the windows

Extra task → Visualization (Conv4 Block 5: DenseNet 121(BM))

GradCam | **Highway** | **Feature Map**

→

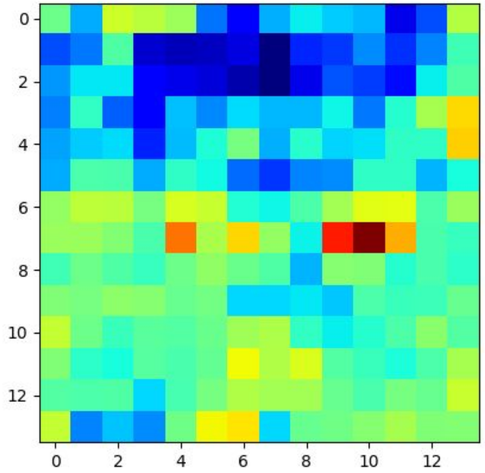


Extra task → Visualization (Last Conv: DenseNet 121(Best model))

GradCam

Highway

Feature Map



Conclusions

- The DenseNet121 does not have many parameters (7M parameters), compared to other SOTA CNNs. So the task of finding a model with fewer parameters with similar performance consisted of removing only one of the four big blocks of the network. The new model has 5M parameters and its performance is very similar to the original.

- Using the new model on the small data set has resulted in worse accuracy, but overall with a large overfit.

- The first step to overcome the over-fitting was to introduce two types of data augmentation: horizontal flipping and zooming.

- Once the validation loss curve stabilized, we considered the use of regulation techniques such as dropouts, early stopping, and automatic lr reduction to further refine the learning curve.
- Good optimizer and regularization improve performance and reduce training time.

- Visualization task: The CNNs try to look for patterns within the image in order to create a featuremap. First Layers look for big patterns(lines..) and last layers is more specialized in figure patterns.