Classification of painter's artwork

Convolutional Neural Network

Michela Capece
Flavia Ferranti

SAPIENZA
UNIVERSITÀ DI ROMA

Vision and perception course

Chosing and ideating project

- The idea behind the theme of the project came from the necessity to re-discover the artistic aspect of life that surrounds us. So instead of let them vanish we decided to create something that could be educationally useful
- Our network classify the painting of ten different artists: Caravaggio, Dalì, Escher, Gentileschi, Gonsalves, Magritte, Monet, Rembrandt, Renoir, Van gogh
- The realization was done using the open source library Tensorflow and its API Kersas

Workflow pipeline

- Our goal was to tune correctly the parameters of a model such that it could be train to map a particular input to an output. To do that we followed some steps:
  - Find the dataset of interest
  - Build the dataset on Tensorflow
  - Build the model
  - Train and test the model
  - Improve it and repeat the process
- To do that we made use of some python libraries such as os, numpy, matplotlib, PIL

Chosing image dataset

- We selected 100 images per painter, without repetition
- Each image were collocated in the right folder
- The size of the image was handled by a specific function: aspect_ratio_resize_smart
- At this point, every image will have a side of 256 and the other side is calculated so that the aspect ratio is unchanged

Creating Dataset

- Building the dataset in Tensorflow require the use of the tf.data API
- The dataset is created taking all properly resized images from the directories with the help of the method : tf.data.Dataset.list_files(*path*)
- In order to have the dataset of the form: (image, label), the name of the folder was exploited
- Get_label, decode_img and process_path are function used to obtain this labeled dataset
  - Get_label takes the name of the folder from the path
  - Decode_image decode the image back to proper format
  - Process_path take the image and calls the other two functions

Training set

- The total amount of images are divided into training and test set, taking 4 over 5 images for the first one
- Each image of the training set are manipulated with the augment function, used to create augmented data; with the following characteristic:
  - Random_crop
  - Random_flip_left_right
  - Random_brighteness
  - Random_saturation
  - Random_contrast
  - Random_hue
- This set is prepared for the training phase with some property of tf.data.Dataset such as .cache, .shuffle, .repeat, .batch, .prefetch

Test set

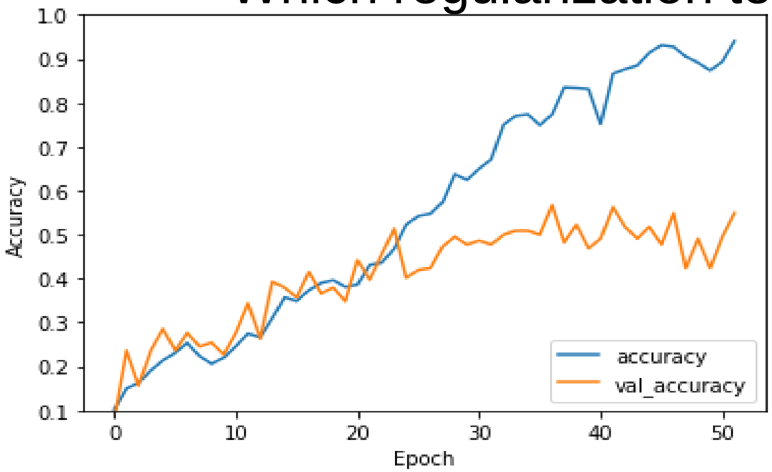
- The test images are centrally cropped
- Another possibility is to resize the whole image
- The augmentation data is not needed for the testing phase
- This set to be prepared for uses the same property listed for the training
- The defined BATCH_SIZE for both training and test is 32

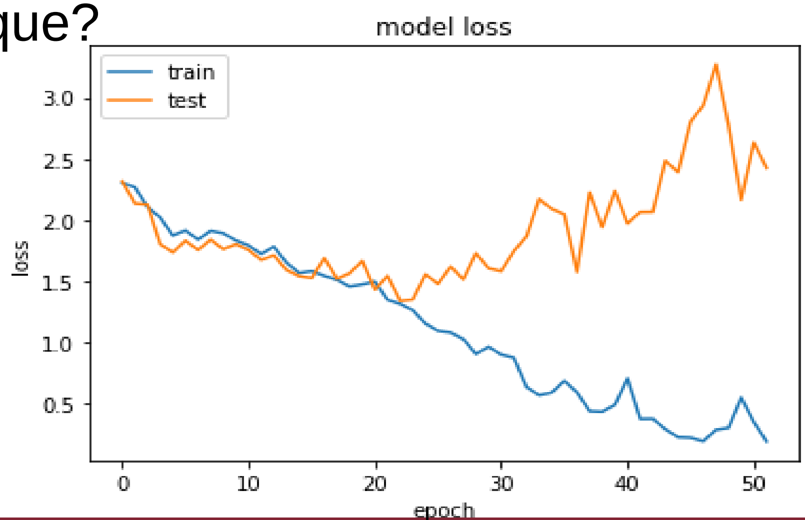
Convolutional Neural Network

- This type if network takes its name from the most critical component of the architecture that is the convolutional layer
- Conv net are divided into 4 components
  - Convolution
  - Non Linearity
  - Pooling
  - Classification
- To construct our own model we had to chose the best parameters to be given at each component

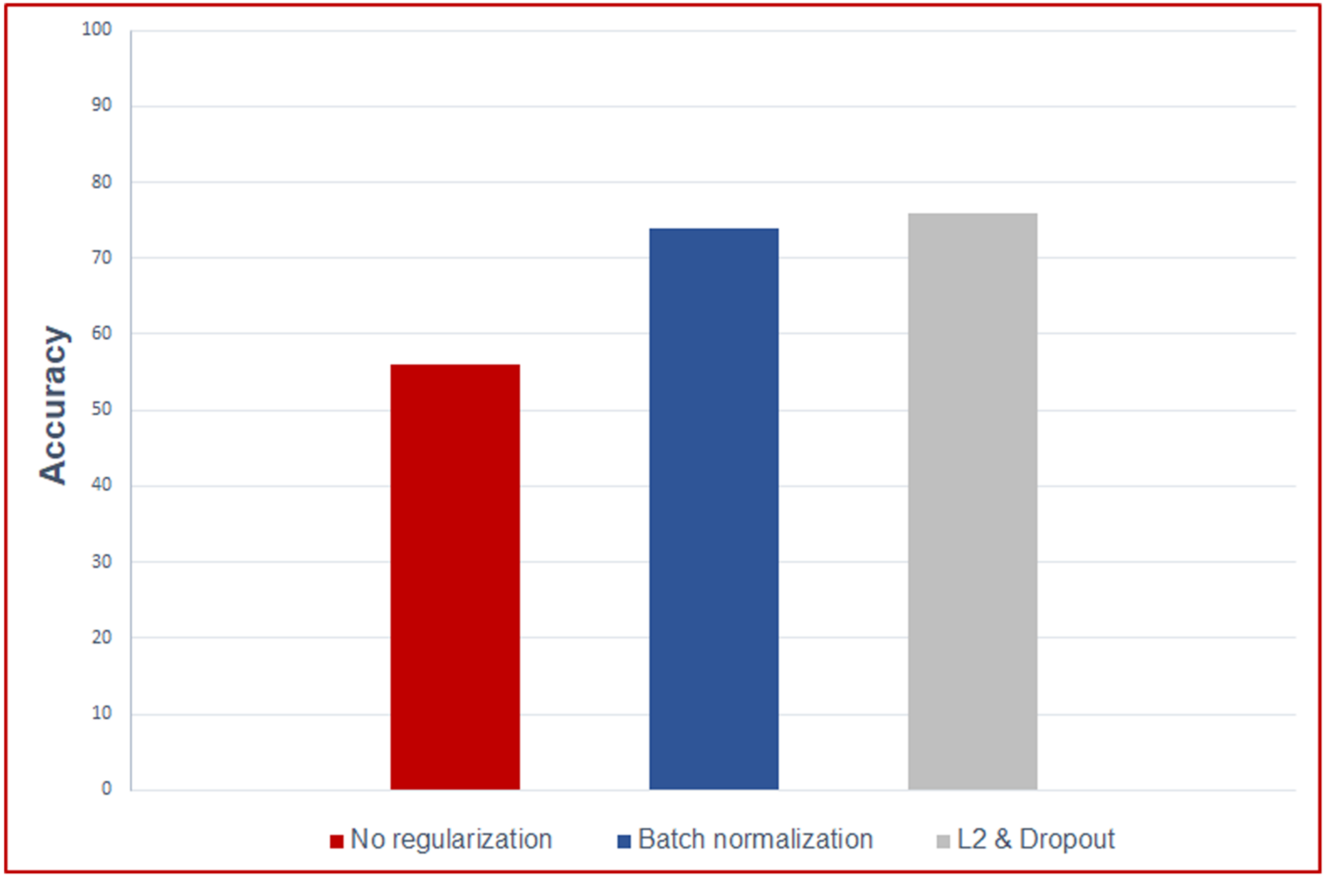
CNN from scratch

- To create a model we had to make several decision:
  - How many filters to use and of which size?
  - How to use padding or stride?
  - Which type of non-linearity?
  - How many layers?
  - What about the activation values?
  - Which regularization technique?





Regularization improvement



CNN architecture

Conv (16,(3x3)) ReLu
Batch normalization
Max pooling ((2x2),(2x2))

↓

Conv (32,(3x3),(2x2)) ReLu
Batch normalization
Conv (64,(3x3)) ReLu
Batch normalization
Max pooling ((2x2),(2x2))

↓

Conv (64,(5x5)) ReLu
Batch normalization
Max pooling ((3x3),(2x2))

↓

Conv (128,(7x7)) ReLu
Batch normalization
Conv (128,(7x7)) ReLu
Batch normalization
Max pooling (3x3)

↓

Conv (256,(5x5)) ReLu
Batch normalization
Max pooling (3x3)
Flatten

↓

Dense (512, L2-regulizer) ReLu
Batch normalization
Dropout (0,5)

↓

Dense (512, L2-regulizer) ReLu
Batch normalization
Dropout (0,5)

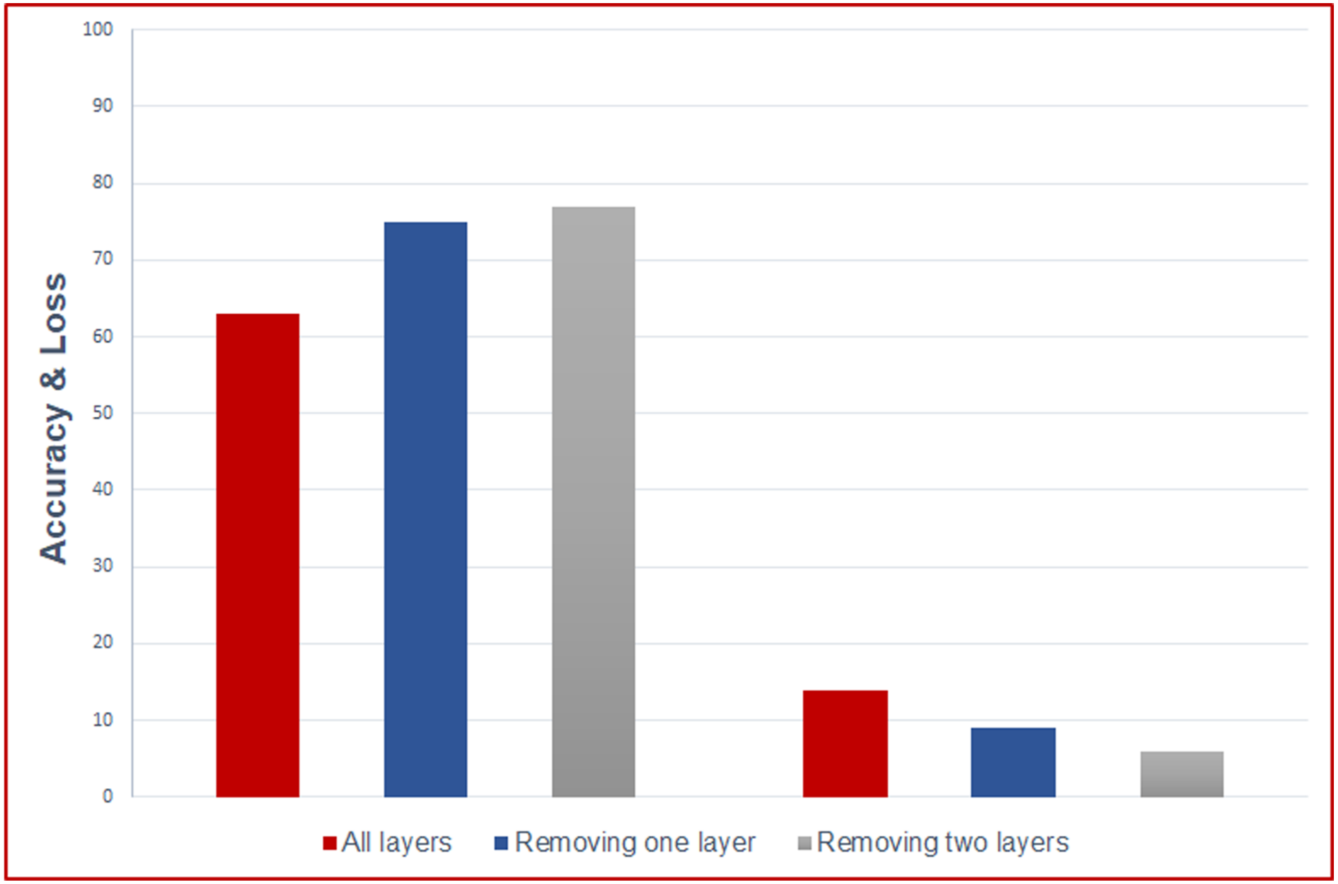
↓

Dense (256, L2-regulizer) ReLu
Batch normalization
Dropout (0,5)

↓

Dense (10) Softmax

CNN from scratch with less data augmentation



Transfer learning

- Method extremely useful because it allows to exploit the knowledge of patterns learned with different problem
- Transfer learning uses pre-trained model
- The pre-trained model was trained on a large benchmark dataset
- Pre-trained model in transfer learning are based on Convolutional Neural Network
- In our case we used two models:
  - ResNet
  - VGG

ResNet as feature extractor

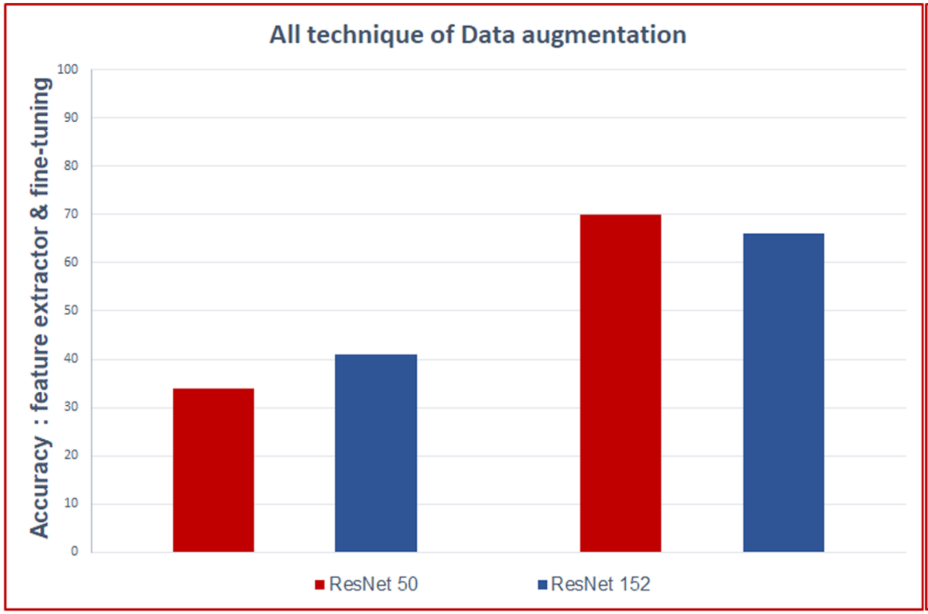
- The Residual Network has a main innovation: the residual module which uses a skip connection. This skip connection exploit the identity paths to connect input and output
- We load ResNet 50 from disk using pre-trained ImageNet weights, without the fully connected layer
- Every ResNet layer is frozen
- Creation of new layers for the prediction
- Training of the latest added layer and evaluation

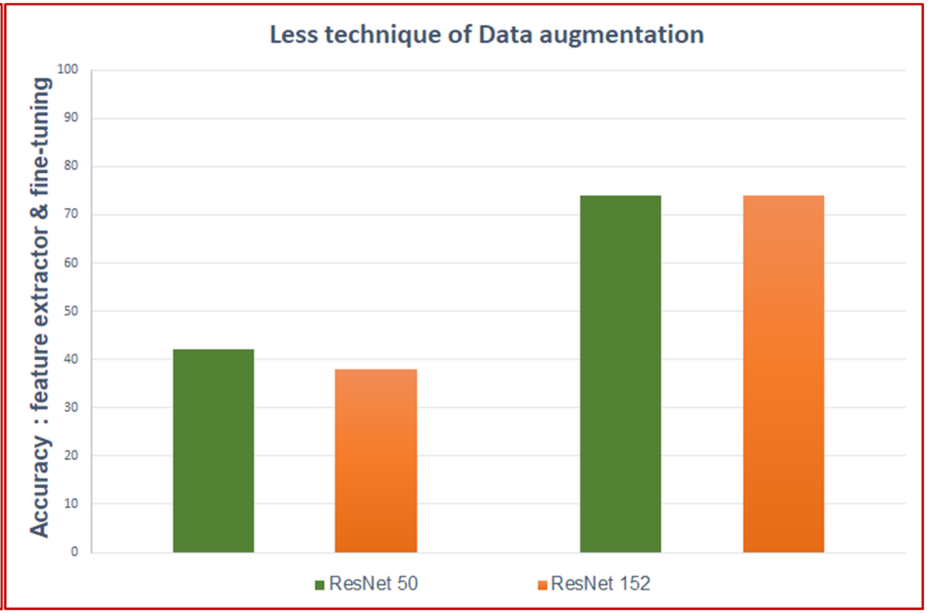
Fine tuning of ResNet model

- To improve the accuracy we made a fine tuning of the whole model
- Each layer can participate to the transfer learning process:
  - The weights are initialized with the pre-trained neural network model
  - The layers are unfrozen
  - During the training process all weights are updated
- The accuracy is higher because the weights are tuned from a generic feature maps to features associated specifically with our dataset

Analyzing ResNet results

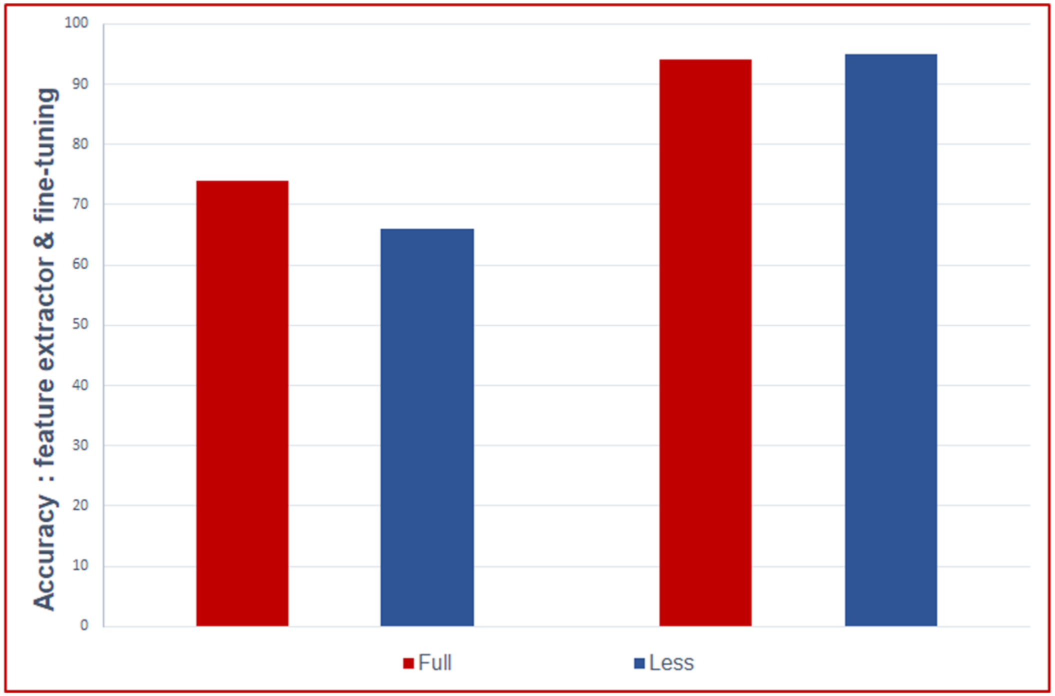
- To improve accuracy we also tried ResNet 152





VGG

- Model composed by 19 layers
- Exploit the power of smaller filters and the stack of various 3x3 conv layer
- VGG was used both as feature extractor and with fine-tuning



Conclusions