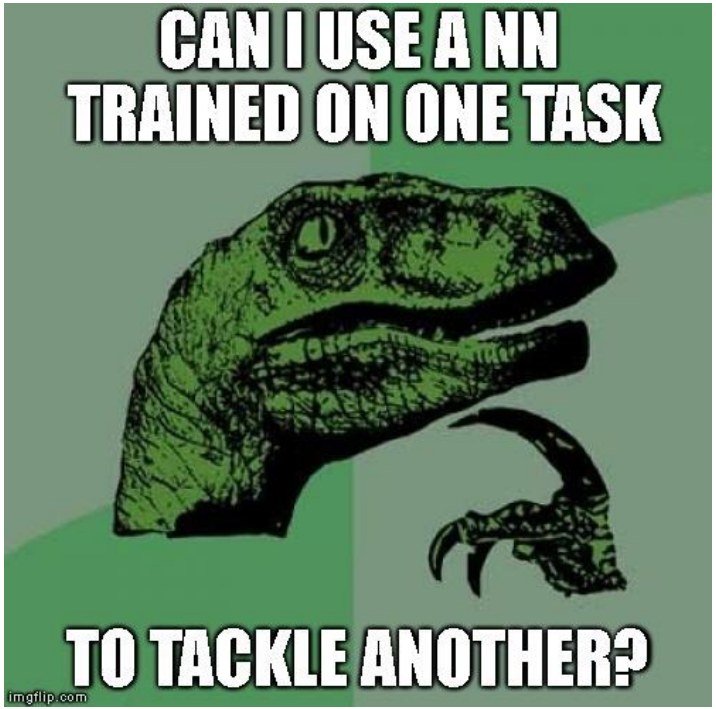
Tutorial 3

A magestic introduction to the 3rd assignment

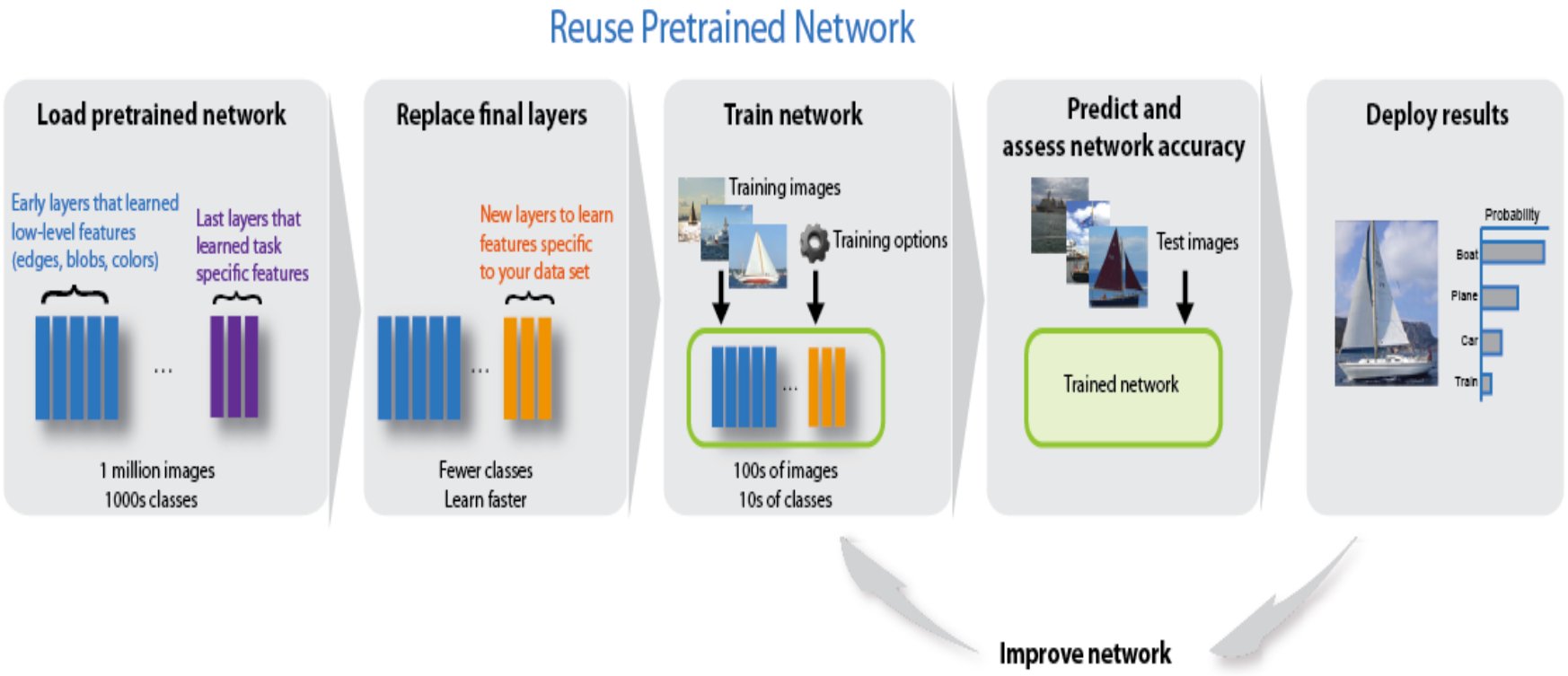
Topics of 3rd assignment



Transfer Learning - Important Q&As

- **What is Transfer Learning?**
  - Transfer learning is a machine learning technique where a model trained on one task is re-purposed on a second related task.
- **Can we do this for every dataset?**
  - No, the original dataset (in which the NN has been trained) and the one in which we want to apply it MUST have similar domains
- **Do we use the NN architecture as it is?**
  - No, early layers are more generic and later layers are original-dataset-specific !
  - We keep the lower layers! Toss the higher ones!
- **So, how we do it?**
  - Load the model pretrained on the original dataset (usually on ImageNet)
  - Remove higher layers (usually the FC ones)
  - Add our dataset-specific layers on top
  - Freeze learning for the lower layers (original) and training the new additions on the new dataset!

Transfer Learning – Basic Concepts



More Q&As - *we will answer them together!*

- **My dataset is small but is similar to the original dataset. Should I fine-tune?**
  - *Answer* :
  - And how about my hyperparameters?

- **My dataset is large and similar to the original dataset. Should I fine-tune or train from scratch?**
  - *Answer:*

- **My dataset is different from the original. Should I finetune?**
  - *Answer:* *(Hint: if it is small we can do something)*

- **How do I decide what to put on - top?**
  - *Answer:*

Task 3.1: Fine-tune a model

- Which model? AlexNet!
- Layers and specs:
  - **Layer 1:**
    - Input: 224 x 224 x 3
    - Number of filters: 96
    - Kernel: 11 x 11 x 3, Stride 4
    - Output: 55 x 55 x 96, Act: ReLU
  - **Layer 2:**
    - Input: 55 x 55 x 96
    - Max - pooling (stride 2) -> Convol
    - Number of filters: 256
    - Kernel: 5 x 5 x 48
    - Output: 27 x 27 x 256
  - **Layers 3, 4 and 5 are similar to 2!**
  - **Layer 6:**
    - Fully Connected: Input -> 13 x 13 x 128
    - Output: 1 x 2048

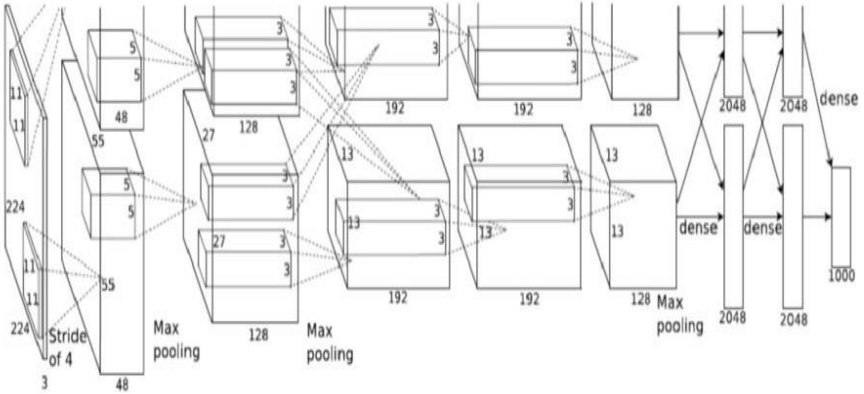


Image Source: http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf

- **Layers 7 and 8 are similar** with 8 outputing 1 x N_classes

Task 3.1: Dataset Description

- WikiArt Dataset
  - 4000 images of paintings
  - 10 different styles (i.e. Baroque, Realism, Expressionism)

Task 3.1: Steps to be followed

- Download the WikiArt dataset (download_wikiart.py file)
  - Be patient, it may take a while…
- Download AlexNet, with the specific weights learned by the network
- Implement the loss function, the optimization process with GD algorithm, and the accuracy obtained
  - Compute the gradients of all variables, and apply GD algorithm
- Fine-tune AlexNet network, by subtracting:
  - The last Fully-Connected (FC) layer
  - The last two final FC layers
- Report accuracy, loss (training & validation) ++ intuition gained
- Epochs: 10 → Training may take a while, gradual improvement though

Image Class Saliency Map Visualization

- Saliency Map: Measures the degree to which each pixel in the image affects the classification score for that image

Image Source:
https://arxiv.org/pdf/1312.6034.pdf.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", ICLR Workshop 2014.

Image Class Saliency Map Visualization

- How do we compute it?

  - Compute the gradient w.r.t image pixels of the un-normalized score corresponding to the correct class

  - Image : (H,W,3) → Gradient : (H,W,3)

  - Obtain absolute value of each of these gradients, and take the maximum value over 3 possible image channels (RGB)

Image Class Saliency Map Visualization

- For every image:
  - Do a forward pass of the image through the network
  - Calculate the scores for every class
  - Enforce the derivative w.r.t. I (i.e. input image) of score vector S at last layer for all classes except class C to be 0, while for C set it to 1
  - Back-propagate this derivative until the start
  - Render the gradients and obtain the Saliency Map

$$S_c(I) = w_c^T I + b_c$$

$$\arg\max_I S_c(I) - \lambda \|I\|_2^2 \qquad w = \left.\frac{\partial S_c}{\partial I}\right|_{I_0}$$

- Network used: VGG_16 (trained on a subset of the ImageNet database)