Neural Network Architectures

Putting everything together

Yordan Darakchiev

Technical Trainer

iordan93@gmail.com

sli.do

#DeepLearning

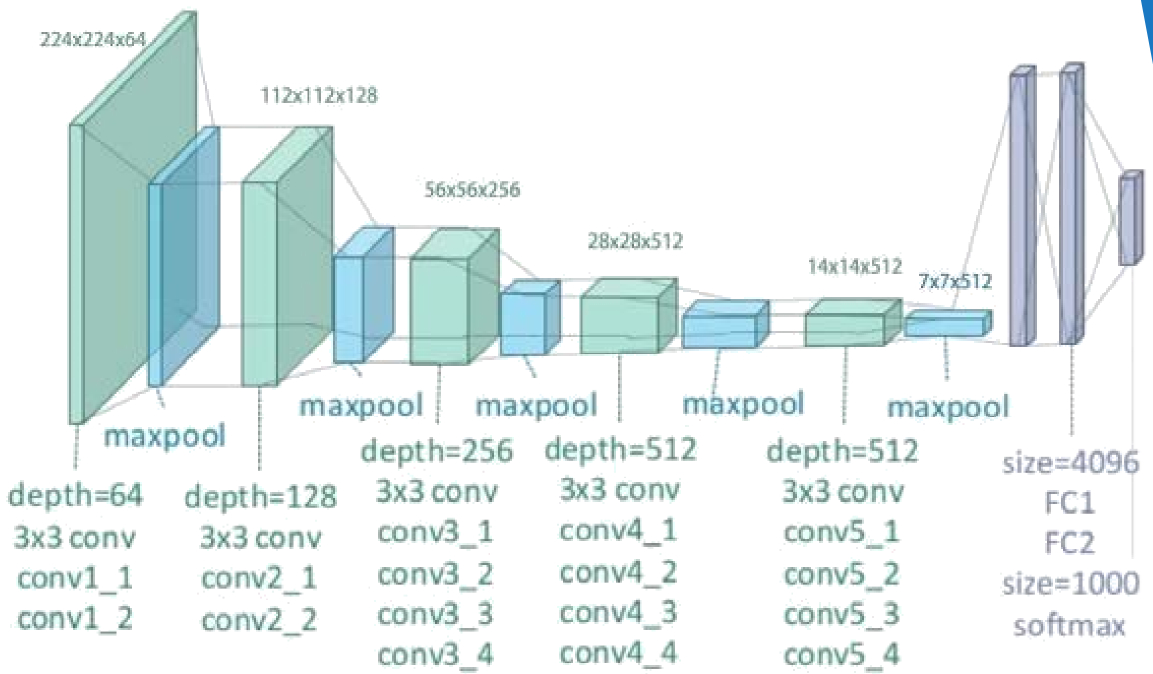
Table of Contents

Architectures

A recap on the popular ones

VGG-19

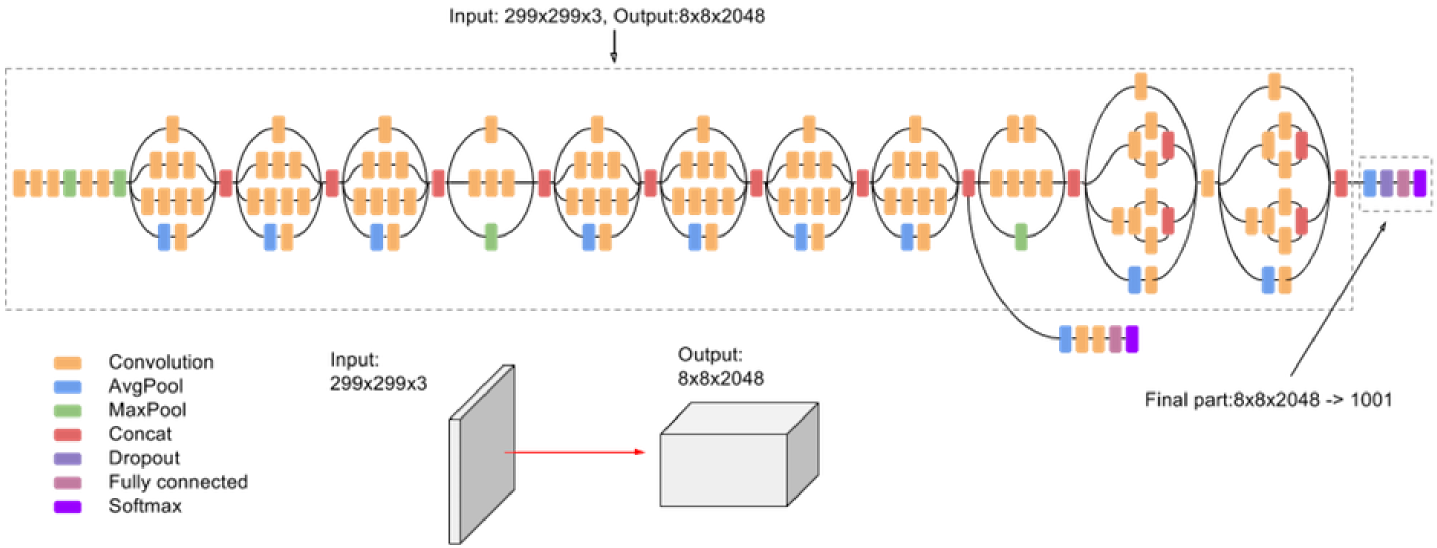
- Input shape: $[n_{img}, 224{,}224{,}3]$
- Output shape: $[n_{img}, 1000]$
- Total params: 143 667 240
- Input flow
  - Read image
  - Resize
  - Preprocess
- Predict
- Decode predictions




```
from tensorflow.keras.applications.vgg19 import \
    preprocess_input, VGG19, decode_predictions
```

Inception v3

- Input shape: $[n_{img}, ?, ?, 3]$
  - Originally $? = 299$
- Output shape: $[n_{img}, 1000]$
- Total params: 23 851 784

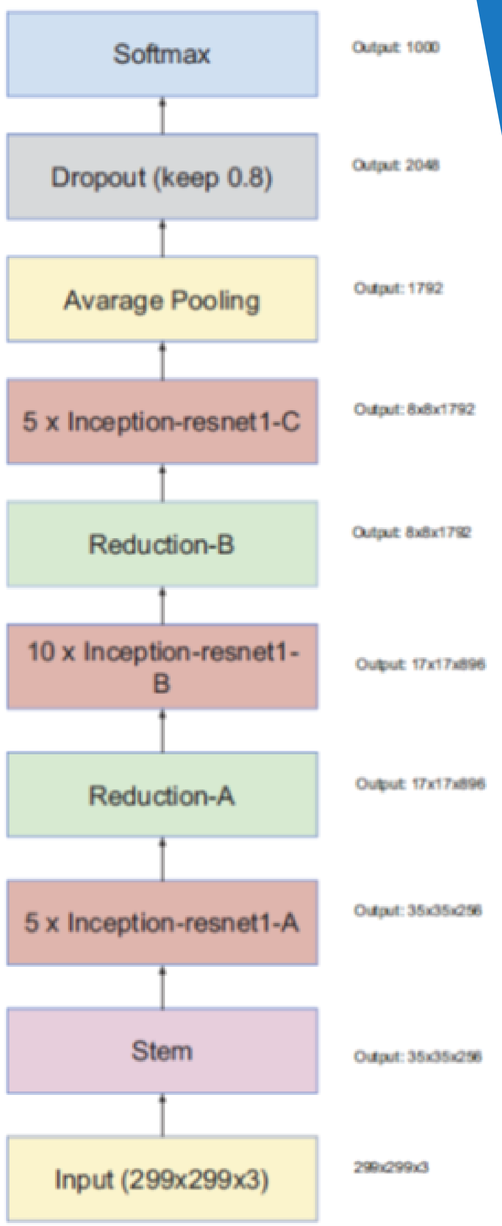



```
from tensorflow.keras.applications.inception_v3 import \
InceptionV3, decode_predictions, preprocess_input
```

Inception-ResNet v2

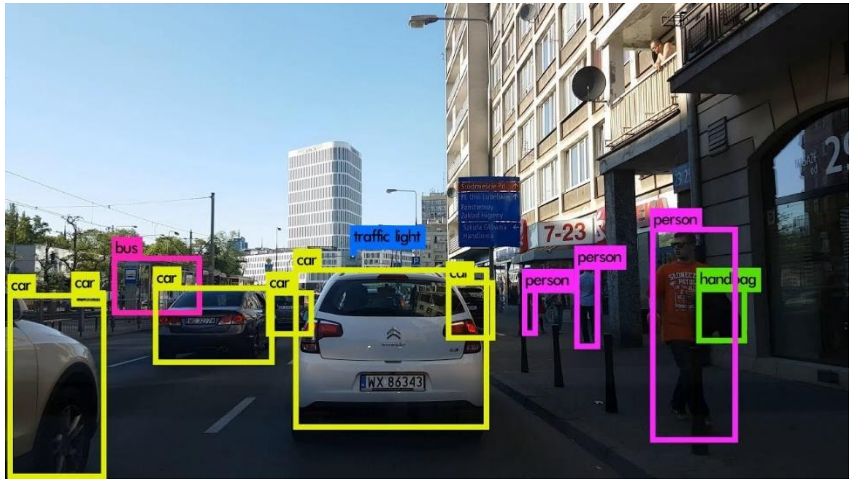
- Input shape: $[n_{img}, ?, ?, 3]$
  - Originally $? = 299$
  - We don't need to resize the image, just apply preprocessing
- Output shape: $[n_{img}, 1000]$
- Total params: 55 873 736

```
from tensorflow.keras.applications. inception_resnet_v2 \
import preprocess_input, InceptionResNetV2, decode_predictions
```



Object Localization

- Input: image; output: bounding box (x, y, w, h)
  - Regression
- Classification and localization
  - Simplest case: 1 object
  - Output a vector: $[p, x, y, w, h, c_1, c_2, \dots, c_k]$
    - $p = 0 \Rightarrow$ no object detected; we don't care about the other numbers
    - $p = 1 \Rightarrow$ object detected; class: $c_1, \dots, c_k$; bounding box x, y, w, h
  - Metrics: usually IoU (or Euclidean distance)
- Implementations: YOLO (You Only Look Once)
  - Also: R-CNN (Region-proposing network)



Transfer Learning

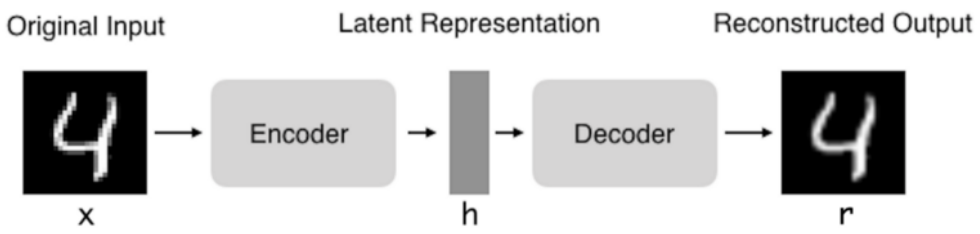
- For a new problem similar to another one that we have already solved, we can reuse the weights
- Prerequisites
  - Small training set on our new model
  - Similar task (i.e. image description)
- Algorithm
  - Remove the last r layers
  - "Freeze" the weights of the remaining layers (`trainable = False`)
    - I.e. use them as a fixed function
  - Add one or more (r') layers
  - Retrain the model (this will update only the last r' layers)

Semi-Supervised Methods

Venturing into unsupervised land

Autoencoders

- NNs which learn to reconstruct their input
  - We care about the latent representation h (like CNNs)



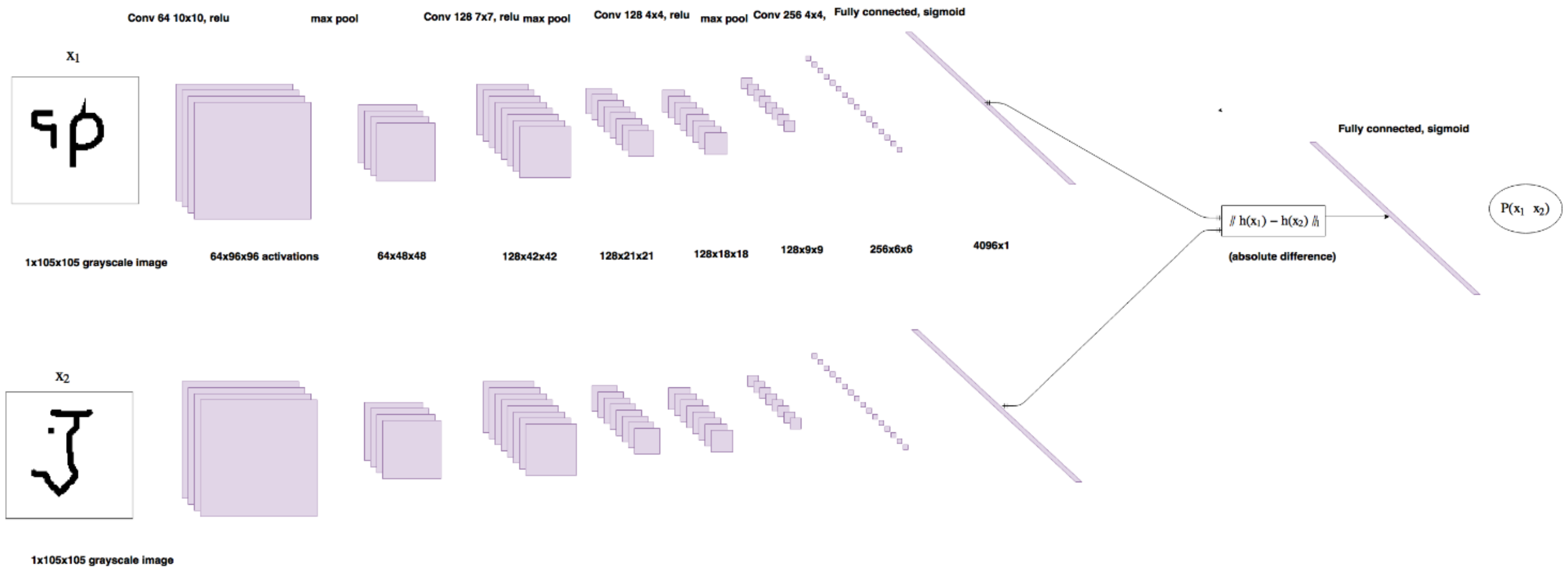
- Encoder / Decoder are simply NNs (can be CNNs, can use multiple layers)
- Main advantages
  - Dimensionality reduction
  - Denoising
- Loss function: difference between x and $\tilde{y}$ (MSE works well)
- Denoising autoencoder
  - Can be used for images, audio, text, etc.
  - Add noise to x to create x_{noise}, compare $\tilde{y}$ to x (**not** to x_{noise})

One-Shot Learning

- Example: Facial recognition
  - Input: image of a face; background data: allowed faces
  - Output: Does the image match any of the allowed faces?
- We aren't allowed to train on many pictures
  - We usually have one picture per person
- This concept generalizes to other multi-class classification tasks
- Solution: Siamese networks
  - Two identical networks receive two images and compute two vectors
  - The distance between the vectors is their (dis)similarity score
  - Dimensionality reduction technique (also similar to clustering)

One-Shot Learning (2)

- Training: **triplet loss** function
  - Input images: anchor a, positive p, negative n
  - Output: vectors e_a, e_p, e_n
  - Loss: low distance from e_a to e_p, high distance from e_a to e_n:
    $$L(a,p,n) = \max\left(d\left(e_a, e_p\right) - d(e_a, e_n) + m, 0\right)$$
    - m – margin (similar to SVMs, allows us to distinguish better)



Novelty Detection

- Similar to one-class SVM ([Chalapathy et al., 2018](#))
- Main idea
  - Assign a confidence score to samples
  - Loss function – minimize distances
- Two approaches
  - If samples have a confidence score (as output)
    ⇒ we can learn even if we have samples of only one class
  - If we don't have a confidence score, we can use similarity measures (i.e. similar embedding vectors)
    - Autoencoder

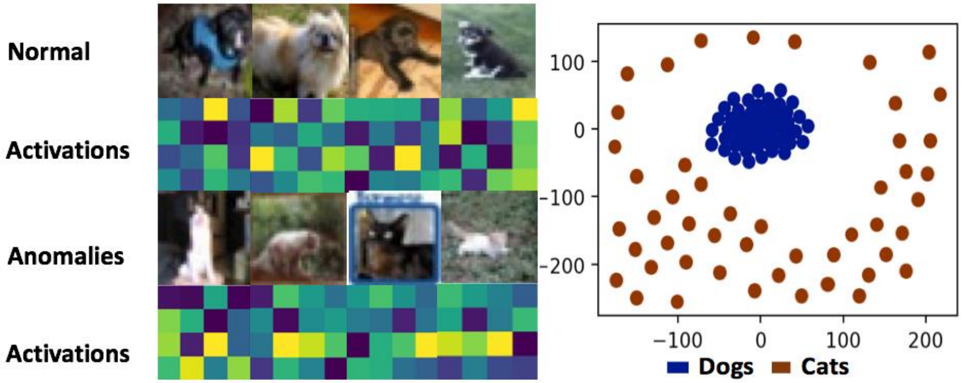
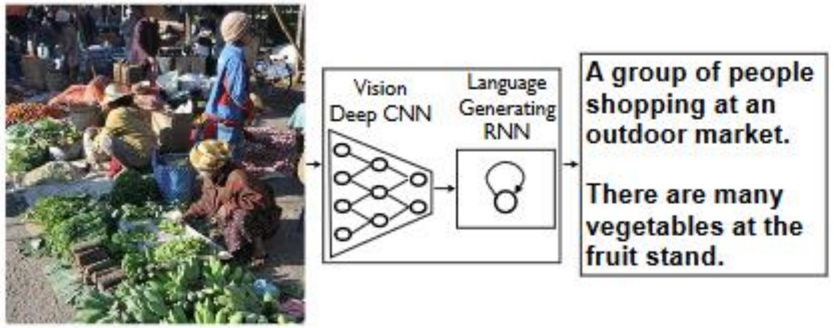


Image Captioning

Describing images to humans

Image Captioning

- [Vinyals et al., 2015](#) ([code](#))
- Similar to machine translation
  - Encoder: CNN instead of RNN
  - Decoder: RNN (LSTM)



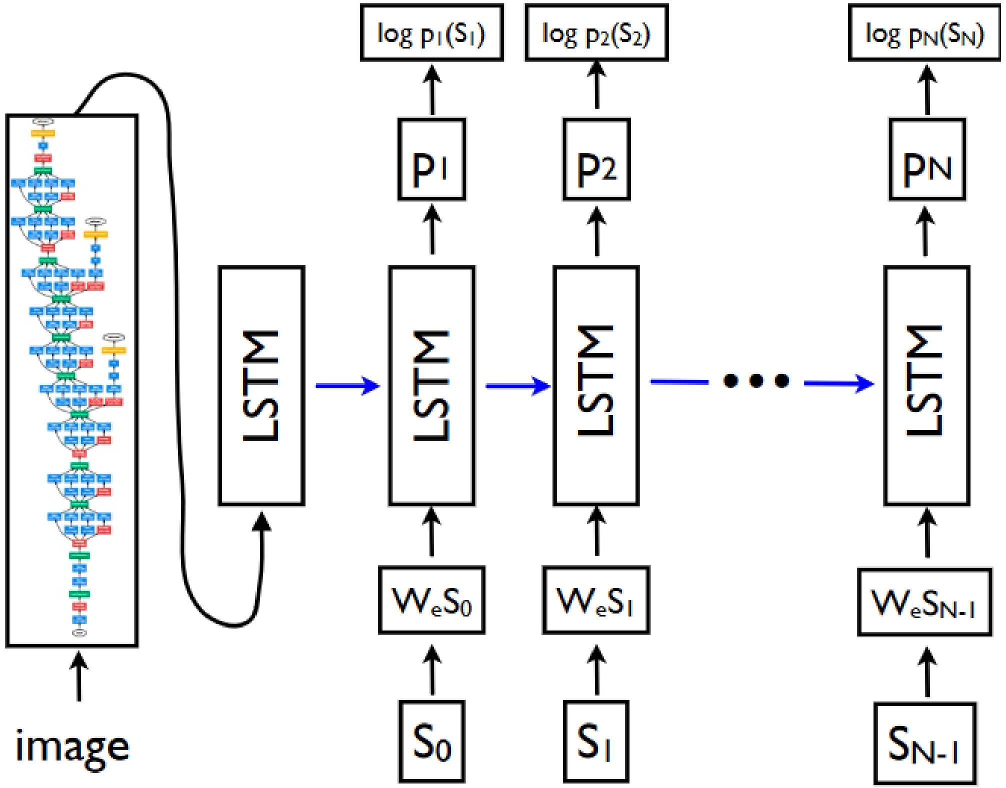
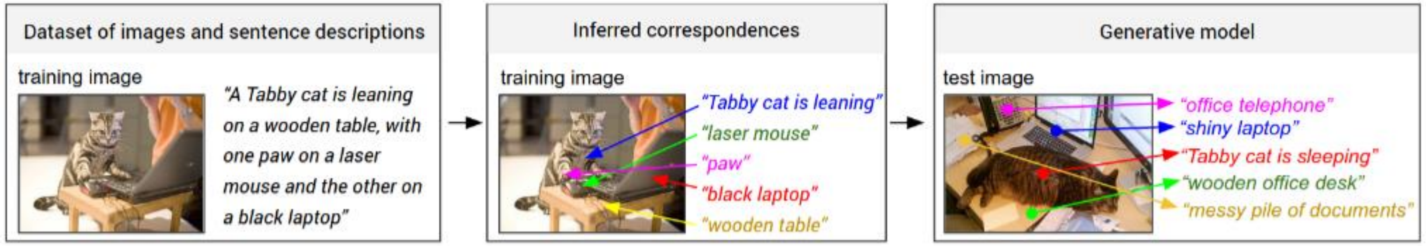


Image Captioning (2)

- Extension: [Karpathy and Li, 2015]()
  - Goal: propose regions and describe them individually
  - R-CNN to get regions
  - Bi-directional RNN to generate sentences



- Both embeddings use the same-dimensional space

Summary

- Architectures: review
- Transfer learning
- Semi-supervised methods
- Image captioning

Questions?