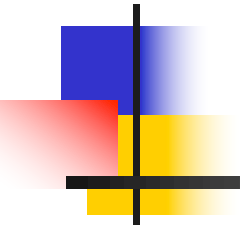
CMSC 510
Regularization Methods for Machine Learning

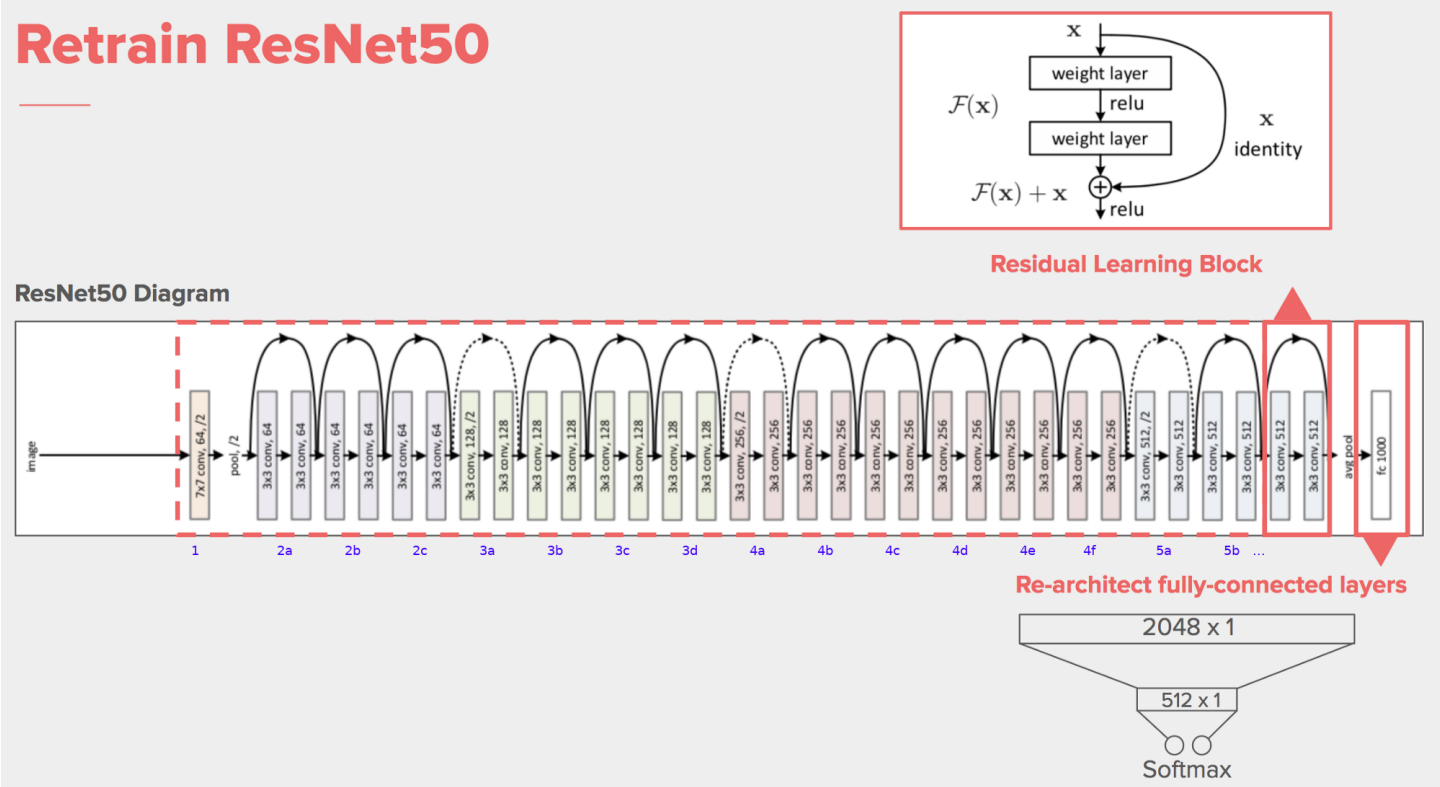


Transfer learning

Instructor:
Dr. Tom Arodz

Models are large

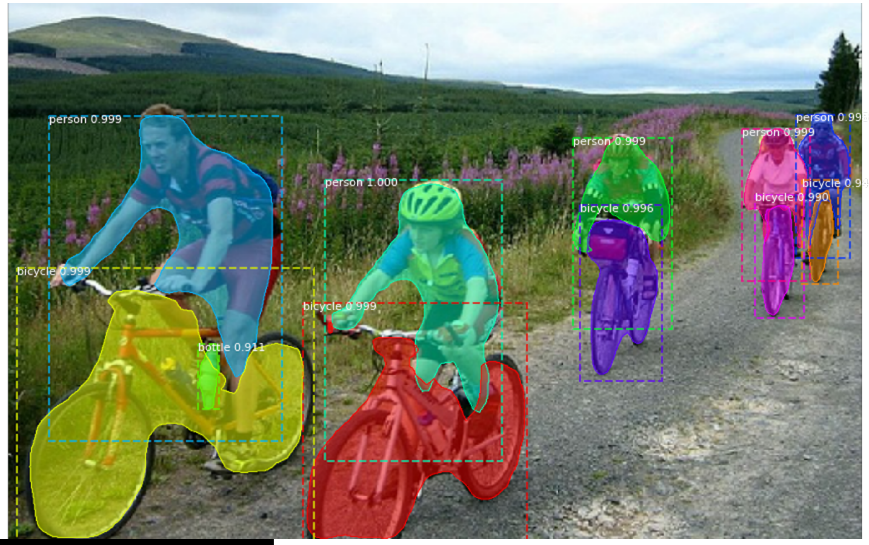
- Currently used deep learning models are very large
- E.g. ResNet50 for image analysis – 23mln weights



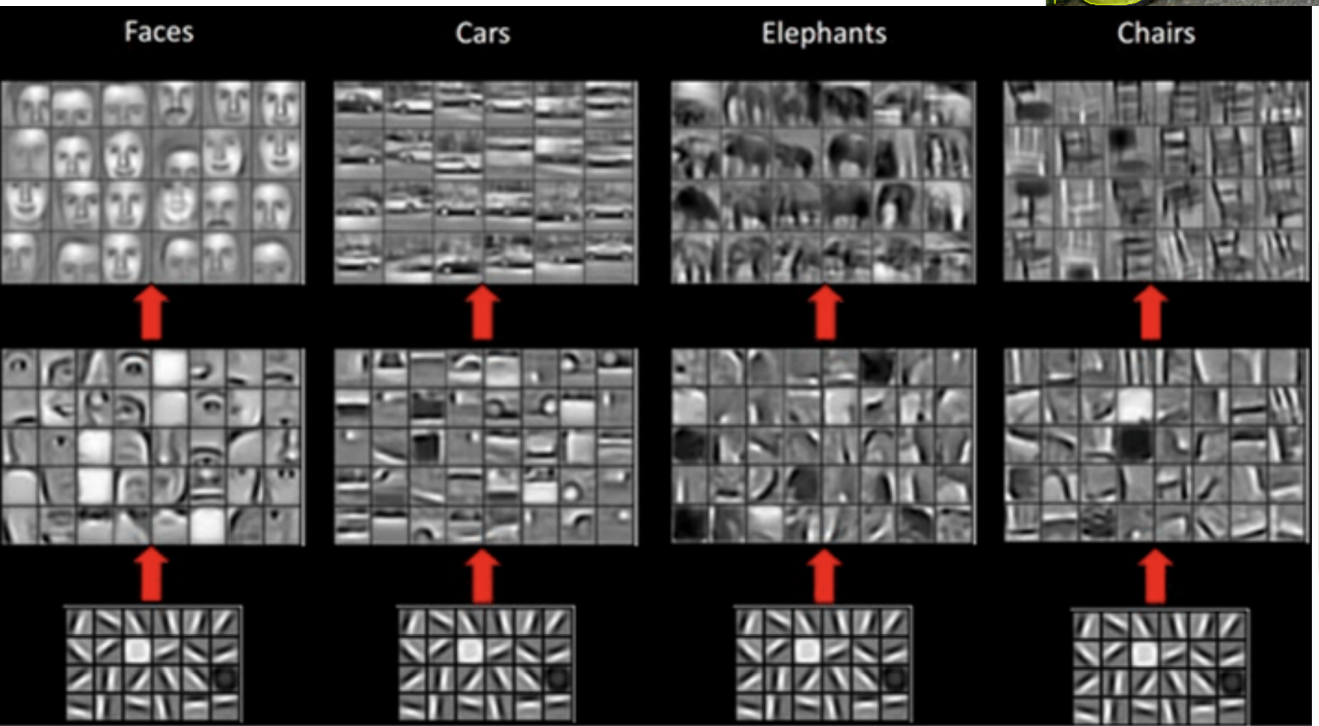
- How to train them on limited data?

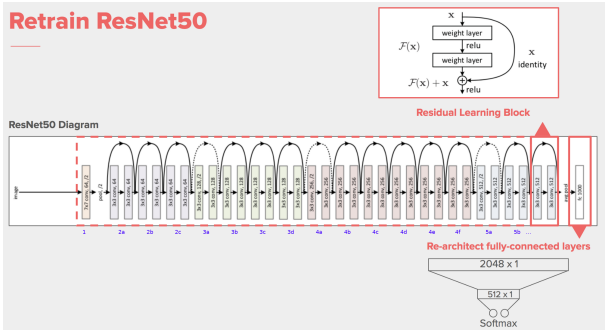
Higher layers – higher representations

- Our task is e.g.
  object detection and segmentation



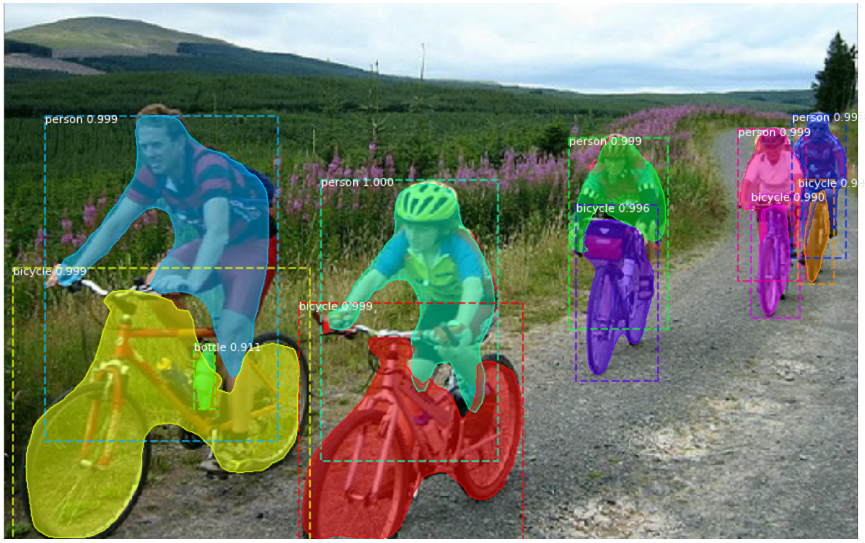
- We can train the network from scratch
  - But it will have to learn all the low-level features about images on earth from scratch





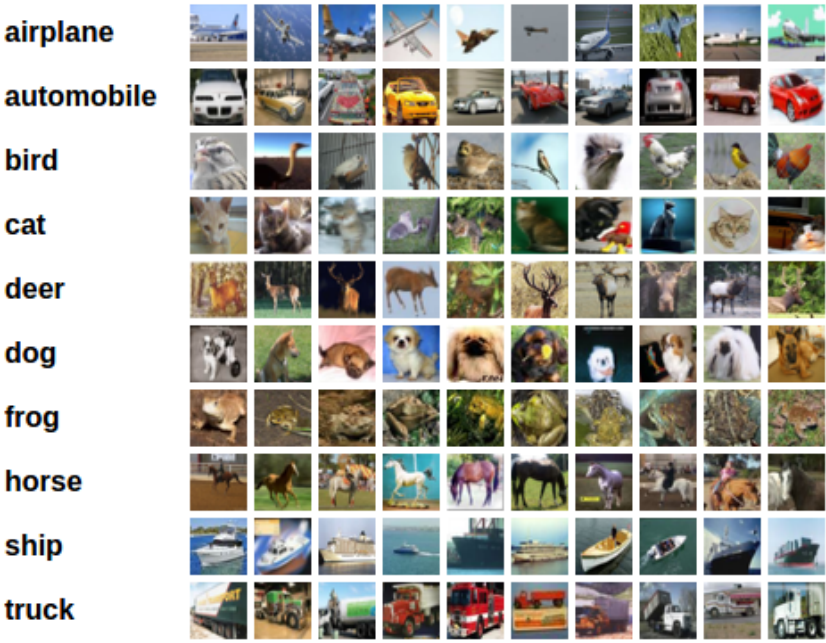
Pre-training / fine-tuning

- Our task is e.g.
  object detection and segmentation



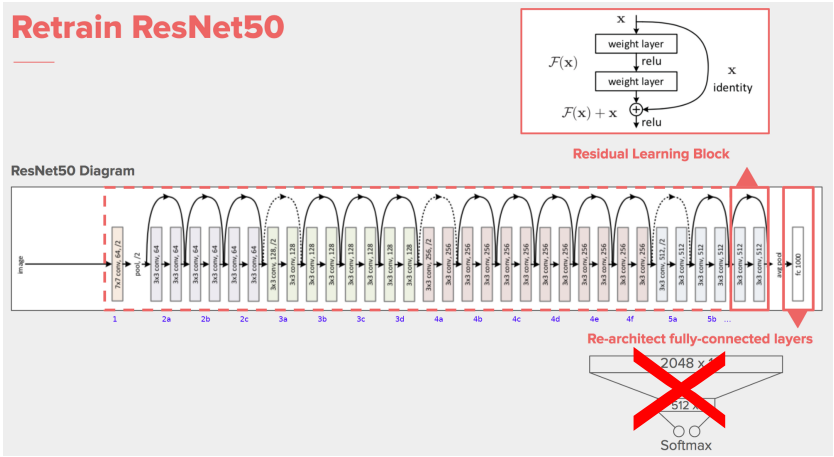
- We can train the network from scratch
  - But it will have to learn all the low-level features about images on earth from scratch

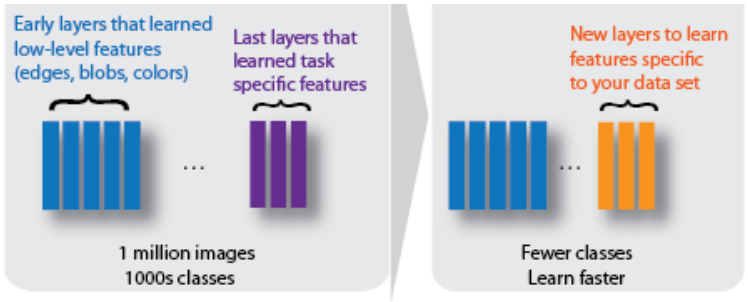
- Alternative – transfer learning:
  - **Pre-train** the network on some related task where a lot of data is available, and we expect similar low-level features/representations
  - Use the trained network instead of a random initialization of weights
  - **Fine-tune** the network towards the desired task



Higher layers – higher representations

- Transfer learning:
  - **Pre-train** on image classification
  - **Fine-tune** for object detection
- The outputs differ:
  - Pre-training: Class of an image
  - Fine-tuning: Pixel membership to objects
- We need to replace the top layers



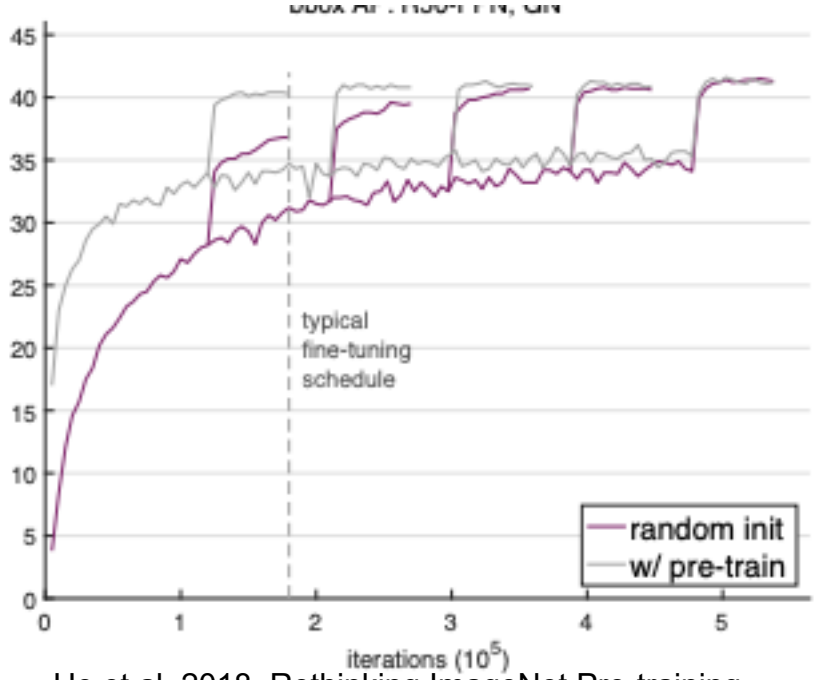


Benefits of pre-training

- Transfer learning:
  - **Pre-train** on image classification
  - **Fine-tune** for object detection

- Pre-training speeds up training and can lead to high-quality networks faster

- Can also make training possible in cases where data for the fine-tuning task is limited

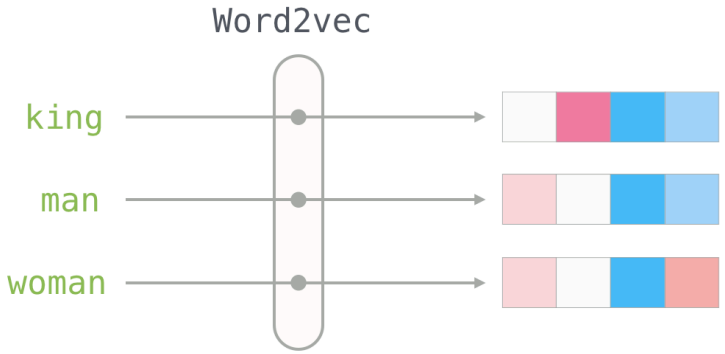


He et al. 2018. Rethinking ImageNet Pre-training

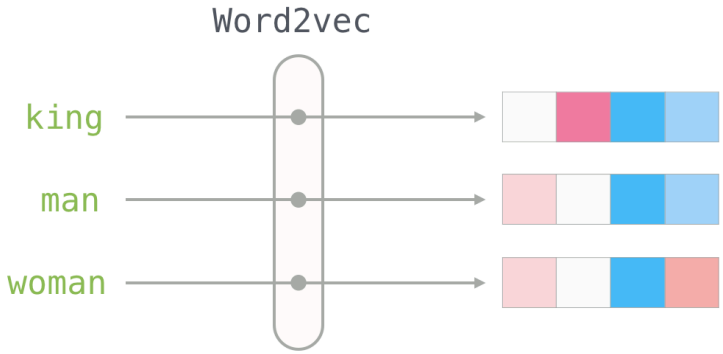
Transfer learning in language understanding

- Pre-training is often used in deep models for natural language understanding
- These models are even larger than for images

- NLP models typically
  - Encode words as vectors- word embeddings
    - Same word – same embedding,
      no matter where / in what sentence
  - Use embeddings from layer k
    to create new "contextualized" embedding
    in layer k+1
    - Same word – different embedding,
      depending on surrounding words
    - How? Attention mechanism



Self-attention



- Use embeddings from layer k to create new "contextualized" embedding in layer k+1
  - Same word – different embedding, depending on surrounding words
  - How? Self-attention mechanism

- Embedding of a word at layer k+1 is* a weighted average of embeddings of all words in the sentence in layer k
  - E.g. narrow street => *street* may "borrow" a bit of meaning from *narrow*

*some additional transformations are added to make it more flexible

- With what weights?
  - Based on attention matrix A
    - Weight of word z in new embedding for word x is $x^T A z$
    - "new x" ≈ $(x^T A z)z + (x^T A s)s + (x^T A t)t$ …
- We need to train matrix A!

Transfer learning in language understanding

- Deep models for natural language understanding are large
  - Large embedding matrix in first layer: e.g. 50,000 x 1024
  - Large attention matrices in subsequent layers: each 1024x1024
  - BERT: 112mln parameters, GPT-2: 1.5bln, GPT-3: 175bln parameters
- Pretraining can use whole Wikipedia to train
- Pre-training on large text dataset
  - Masked language model
    - Train to predict what the missing word in sentence is
  - other, e.g.:
    - Replace some words using a smaller model and classify sounds as original or replaced
- Fine-tuning:
  - Training the network with weights taken from the pre-trained model on a downstream task – e.g. sentiment analysis