Problems in Applying Deep Learning

When applying deep learning in practice:

- Usually, **deep network** architectures are required for **complex tasks**.
- They have many parameters and **need large labeled datasets** to obtain good generalization.
- Generating large "labeled" dataset for your task is difficult.
  - **Time consuming**: You may want to complete the project withing tight deadlines.
  - **Expensive**: Annotating a large dataset is expensive (infrastructure and manpower).
  - **Not possible**: e.g. in some medical imaging tasks, it is not ethical to put a healthy person through imaging or prevalence can be low.

Deep learning can only be applied when large datasets are available.

Problems in Applying Deep Learning

When applying deep learning in practice:

- Usually, **deep network** architectures are required for **complex tasks**.
- They have many parameters and **need large labeled datasets** to obtain good generalization.
- Generating large "labeled" dataset for your task is difficult.
    - **Time consuming**: You may want to complete the project withing tight deadlines.
    - **Expensive**: Annotating a large dataset is expensive (infrastructure and manpower).
    - **Not possible**: e.g. in some medical imaging tasks, it is not ethical to put a healthy person through imaging or prevalence can be low.

~~Deep learning can only be applied when large labeled datasets are available.~~

Not correct! There are alternatives.

Objective of the lecture

Explore practical approaches used in deep learning that can handle limited data availability:

- Relatively small labeled dataset only: **Transfer learning**.
- Large unlabelled data: **Self supervised learning**.

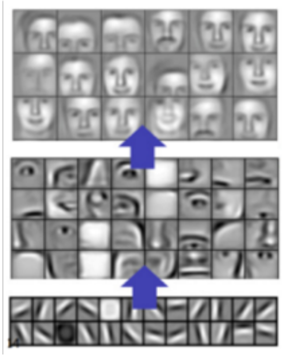


Image from: Paper

Learned features from a face recognition task.

Observation:

- Layers towards the bottom (close to input) learn *general features*.
- Layers towards the top (close to output) learn task *specific features*.

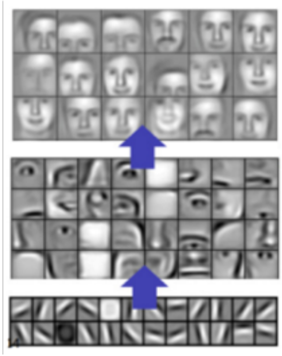


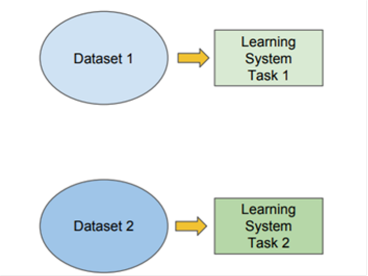
Image from: Paper

Learned features from a face recognition task.

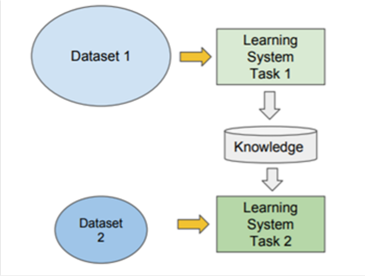
Observation:

- Layers towards the bottom (close to input) learn *general features*.
- Layers towards the top (close to output) learn task *specific features*.

Can we take the learned features on one task and apply them to a new task?



Traditional



Transfer Learning

Assume you are designing a "Pet feeding machine" that dispenses the appropriate food type based on weather a cat or a dog is nearby.

You have decided to use vision to identify if a cat or a dog is nearby.

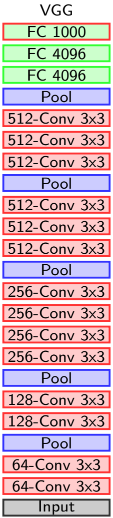

Cats vs Dogs classification can be complex as there are many types of cats/dogs.

Can collect a small dataset with labels. Training dataset size: 2000 images from both classes.

Model Training Strategy

- Find a very large dataset that has similar data, train a big CNN there (or take pre-trained models).
- Transfer what is learned to your dataset.

Why would this word? Features in the bottom layers are general. Knowledge gained will transfer to other tasks.



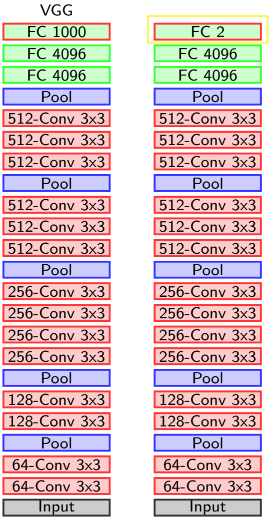
Step 1: Select a related task (proxy task) that has large dataset. e.g. ImageNet.

Step 2: Train a selected base model on the proxy task or download the pre-trained model. e.g. VGG/ResNet.

We can assume that the top most layer is very specific to the ImageNet task (not very related to our cats vs dog). The bottom layers are general.

$\ast$ This is not the best example because cats and dogs are categories in ImageNet. The process will work even when the categories are not directly in ImageNet (E.g. Gaze direction classification in face images.)

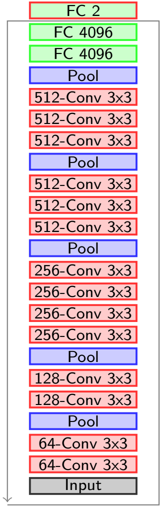
Transfer Learning for Image Classification Tasks



Step 3: Identify the feature layer you want to use for your task. Usually the **bottleneck layer** (the one before the final classification layer).

Step 4: Remove the layers above the bottleneck layer and reconfigure to your task.

- New top part is called the **head**.
- Remaining parts from the pre-trained model is called the **base model**.



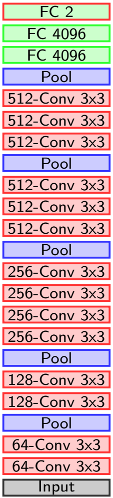
Step 5: Freeze all the layer below (and including) bottleneck layer (weights in base model).

Step 6: Train the network on the new task. E.g Cats vs Dogs.

Step 7: (optional) Fine-tuning; Unfreeze "some" top layers in the frozen base model and train the network for some more epoch on the new task. This will train both head and base model weights.

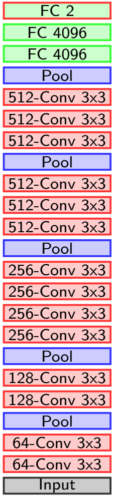
Now you have a trained model for the task. We will do an example in the lab.

Important Considerations



- The **input normalization and pre-processing** has to match with the techniques employed in training the original model.
- Should use a **smaller learning rate when fine-tuning** to avoid forgetting the pre-learned features. Over-fitting to new task.
- Should not train the head and the base model weights together without first tuning the head with frozen base model.

Important Considerations



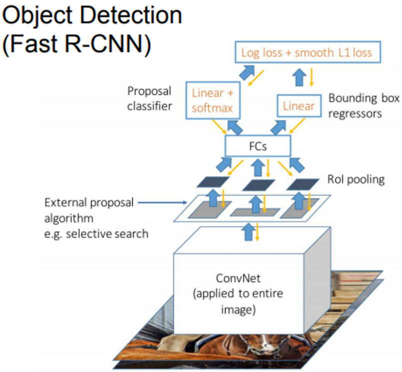
	Similar datasets	**Different datasets**
Relatively Small Dataset	Train the classifier on top	You are in trouble. Get more data
Relatively Large dataset	Fine tune Few more layers	Fine tune a larger number of layers

Transfer learning for images: summary

RMIT UNIVERSITY

1. Select a related proxy task that has large dataset (or pre-trained model).
2. Train a selected base model on the proxy task or download the pre-trained model.
3. Identify the feature layer you want to use for your task.
4. Remove the layers above the bottleneck layer and reconfigure to your task (attach a head).
5. Freeze all the layer in base model.
6. Train the network on the new task.
7. (optional) Unfreeze some top layers in the base model and fine-tune.

Transfer learning with CNNs



Object Detection

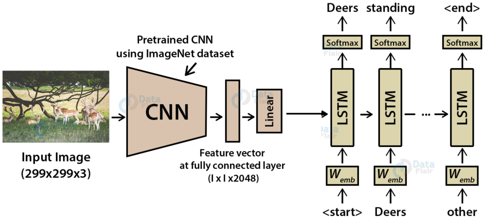


Image Captioning

Nowadays Transfer learning is the norm.