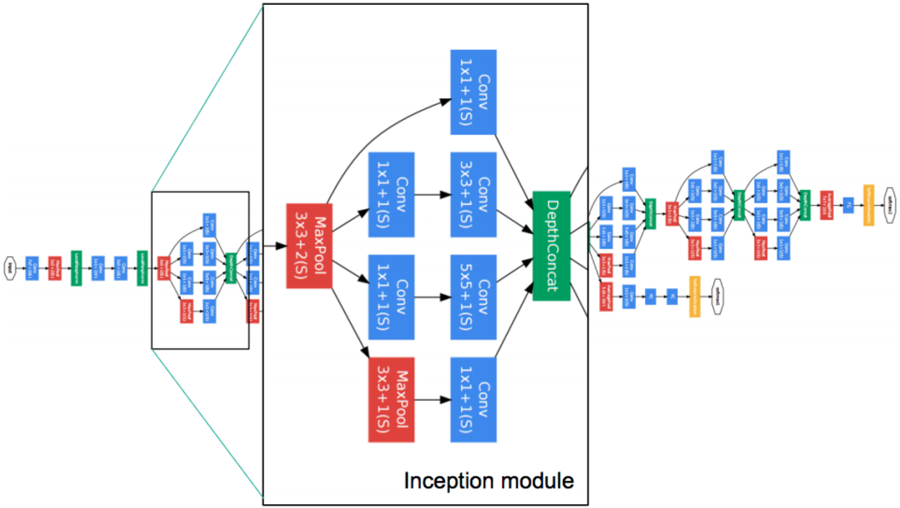
Deep Learning

Vazgen Mikayelyan

YSU, Krisp

November 18, 2020

GoogLeNet/Inception v1 (2014)



Receptive Field

Receptive Field

Let we have a convolutional neural network with consequent layers $F_0, F_1, \ldots, F_D$ (F_0, F_D are inputs and outputs of the network respectively).

Receptive Field

Let we have a convolutional neural network with consequent layers $F_0, F_1, \ldots, F_D$ (F_0, F_D are inputs and outputs of the network respectively).

Receptive Field

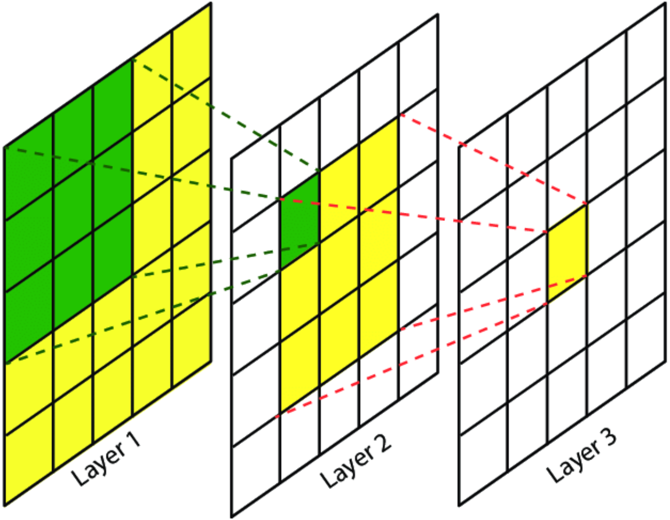
The Receptive Field of a layer F_k with respect to the layer $F_m\,(m < k)$ is the maximal region in the layer F_m, each element of which is contributed in forming one of pixels in F_k, considering only convolutional layers as contribution.

Let we have a convolutional neural network with consequent layers $F_0, F_1, \ldots, F_D$ (F_0, F_D are inputs and outputs of the network respectively).

Receptive Field

The Receptive Field of a layer F_k with respect to the layer F_m $(m < k)$ is the maximal region in the layer F_m, each element of which is contributed in forming one of pixels in F_k, considering only convolutional layers as contribution. By receptive field of a network we mean the receptive field of its output with respect to its input.

Receptive Field



Convolution Factorization

The concept of convolution factorization was introduced in the Inception-V3 architecture.

Convolution Factorization

The concept of convolution factorization was introduced in the Inception-V3 architecture. The idea is the following: in some places

- replace 5×5 convolution layers with two 3×3 convolution layers,

Convolution Factorization

The concept of convolution factorization was introduced in the Inception-V3 architecture. The idea is the following: in some places

- replace 5×5 convolution layers with two 3×3 convolution layers,
- replace $n \times n$ convolution layers with asymmetric consequent convolution layers of sizes $n \times 1$ and $1 \times n$.

Convolution Factorization

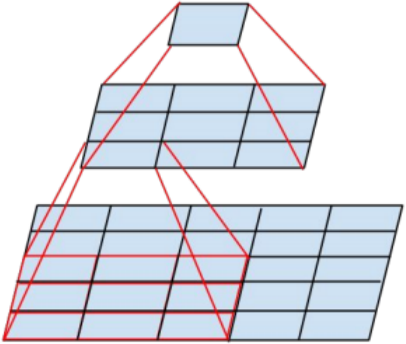


Figure 1. Mini-network replacing the 5×5 convolutions.

Convolution Factorization

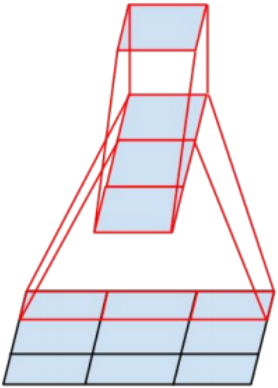


Figure 3. Mini-network replacing the 3×3 convolutions. The lower layer of this network consists of a 3×1 convolution with 3 output units.

Inceptions

- GoogLeNet/Inception v1 (2014) has approximately 5M parameters.

- GoogLeNet/Inception v1 (2014) has approximately 5M parameters.
- Accuracies on ImageNet: Top1=69.8%, Top5=89.9%.

- GoogLeNet/Inception v1 (2014) has approximately 5M parameters.
- Accuracies on ImageNet: Top1=69.8%, Top5=89.9%.
- Inception v2 (2015) has approximately 11.2M parameters.

Inceptions

- GoogLeNet/Inception v1 (2014) has approximately 5M parameters.
- Accuracies on ImageNet: Top1=69.8%, Top5=89.9%.
- Inception v2 (2015) has approximately 11.2M parameters.
- Accuracies on ImageNet: Top1=74.8%, Top5=92.2%.

- GoogLeNet/Inception v1 (2014) has approximately 5M parameters.
- Accuracies on ImageNet: Top1=69.8%, Top5=89.9%.
- Inception v2 (2015) has approximately 11.2M parameters.
- Accuracies on ImageNet: Top1=74.8%, Top5=92.2%.
- Inception v3 (2015) has approximately 23.8M parameters.

Inceptions

- GoogLeNet/Inception v1 (2014) has approximately 5M parameters.
- Accuracies on ImageNet: Top1=69.8%, Top5=89.9%.
- Inception v2 (2015) has approximately 11.2M parameters.
- Accuracies on ImageNet: Top1=74.8%, Top5=92.2%.
- Inception v3 (2015) has approximately 23.8M parameters.
- Accuracies on ImageNet: Top1=78.8%, Top5=94.4%.

Inceptions

- GoogLeNet/Inception v1 (2014) has approximately 5M parameters.
- Accuracies on ImageNet: Top1=69.8%, Top5=89.9%.
- Inception v2 (2015) has approximately 11.2M parameters.
- Accuracies on ImageNet: Top1=74.8%, Top5=92.2%.
- Inception v3 (2015) has approximately 23.8M parameters.
- Accuracies on ImageNet: Top1=78.8%, Top5=94.4%.
- Inception v4/Inception-ResNet (2016) has approximately 55.8M parameters.

Inceptions

- GoogLeNet/Inception v1 (2014) has approximately 5M parameters.
- Accuracies on ImageNet: Top1=69.8%, Top5=89.9%.
- Inception v2 (2015) has approximately 11.2M parameters.
- Accuracies on ImageNet: Top1=74.8%, Top5=92.2%.
- Inception v3 (2015) has approximately 23.8M parameters.
- Accuracies on ImageNet: Top1=78.8%, Top5=94.4%.
- Inception v4/Inception-ResNet (2016) has approximately 55.8M parameters.
- Accuracies on ImageNet: Top1=80.1%, Top5=95.1%.

Inceptions

- GoogLeNet/Inception v1 (2014) has approximately 5M parameters.
- Accuracies on ImageNet: Top1=69.8%, Top5=89.9%.
- Inception v2 (2015) has approximately 11.2M parameters.
- Accuracies on ImageNet: Top1=74.8%, Top5=92.2%.
- Inception v3 (2015) has approximately 23.8M parameters.
- Accuracies on ImageNet: Top1=78.8%, Top5=94.4%.
- Inception v4/Inception-ResNet (2016) has approximately 55.8M parameters.
- Accuracies on ImageNet: Top1=80.1%, Top5=95.1%.
- The best result on ImageNet: Top1=88.5%, Top5=98.7% with 480M parameters.

Outline

Problems of deep learning

- It is difficult to find datasets of huge sizes, and it is way too costly to prepare such datasets.

Problems of deep learning

- It is difficult to find datasets of huge sizes, and it is way too costly to prepare such datasets.
- The high cost of GPUs needed to run advanced deep learning algorithms.

Problems of deep learning

- It is difficult to find datasets of huge sizes, and it is way too costly to prepare such datasets.
- The high cost of GPUs needed to run advanced deep learning algorithms.
- Training takes a long time.

Idea of transfer learning

Instead of training a deep network from scratch for your task:

Idea of transfer learning

Instead of training a deep network from scratch for your task:

- Take a network trained on a different domain for a different **source task**.

Idea of transfer learning

Instead of training a deep network from scratch for your task:

- Take a network trained on a different domain for a different **source task**.
- Adapt it for your domain and your **target task**.

Idea of transfer learning

Instead of training a deep network from scratch for your task:

- Take a network trained on a different domain for a different **source task**.
- Adapt it for your domain and your **target task**.

Variations:

- Same domain, different task.
- Different domain, same task.

Idea of transfer learning

When transfer learning makes sense?

Idea of transfer learning

When transfer learning makes sense?

- Task A and B have the same input x.

Idea of transfer learning

When transfer learning makes sense?

- Task A and B have the same input x.
- You have a lot of more data for Task A than Task B.

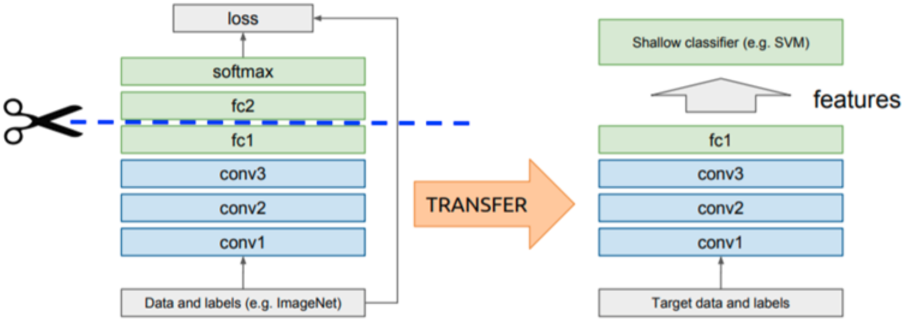
When transfer learning makes sense?

- Task A and B have the same input x.
- You have a lot of more data for Task A than Task B.
- Low level features from A could be helpful for learning B.

Fixed feature extractor

For example take a NN pretrained on some big dataset, remove the last fully-connected layer, then treat the rest of the NN as a fixed feature extractor for the new dataset. Then train a classifier (e.g. Linear SVM or Softmax classifier) for the new dataset.

Fixed feature extractor



Fine-tuning

This strategy is to not only replace and retrain the classifier on top of the deep net on the new dataset, but to also fine-tune the weights of the pretrained network by continuing the backpropagation.

This strategy is to not only replace and retrain the classifier on top of the deep net on the new dataset, but to also fine-tune the weights of the pretrained network by continuing the backpropagation.
Variations:

Fine-tuning

This strategy is to not only replace and retrain the classifier on top of the deep net on the new dataset, but to also fine-tune the weights of the pretrained network by continuing the backpropagation.

Variations:

- Fine-tune all the layers of the network.

Fine-tuning

This strategy is to not only replace and retrain the classifier on top of the deep net on the new dataset, but to also fine-tune the weights of the pretrained network by continuing the backpropagation.

Variations:

- Fine-tune all the layers of the network.
- Fine-tune only some higher-level portion of the network.

Fine-tuning

This strategy is to not only replace and retrain the classifier on top of the deep net on the new dataset, but to also fine-tune the weights of the pretrained network by continuing the backpropagation.

Variations:

- Fine-tune all the layers of the network.
- Fine-tune only some higher-level portion of the network.
- Fine-tune all the layers of the network with different learning rates.

Fine-tuning

This strategy is to not only replace and retrain the classifier on top of the deep net on the new dataset, but to also fine-tune the weights of the pretrained network by continuing the backpropagation.

Variations:

- Fine-tune all the layers of the network.
- Fine-tune only some higher-level portion of the network.
- Fine-tune all the layers of the network with different learning rates.

The last two points are motivated by the observation that the earlier features of the network contain more generic features.

When and how to fine-tune?

How do you decide what type of transfer learning you should perform on a new dataset?

When and how to fine-tune?

How do you decide what type of transfer learning you should perform on a new dataset? This is a function of several factors, but the two most important ones are the size of the new dataset, and its similarity to the original dataset.

When and how to fine-tune?

How do you decide what type of transfer learning you should perform on a new dataset? This is a function of several factors, but the two most important ones are the size of the new dataset, and its similarity to the original dataset.

Scenarios:

- New dataset is large and very different from the original dataset.

When and how to fine-tune?

How do you decide what type of transfer learning you should perform on a new dataset? This is a function of several factors, but the two most important ones are the size of the new dataset, and its similarity to the original dataset.

Scenarios:

- New dataset is large and very different from the original dataset.

  Train from scratch the whole network.

When and how to fine-tune?

How do you decide what type of transfer learning you should perform on a new dataset? This is a function of several factors, but the two most important ones are the size of the new dataset, and its similarity to the original dataset.

Scenarios:

- New dataset is large and very different from the original dataset.

  Train from scratch the whole network.
- New dataset is large and similar to the original dataset.

When and how to fine-tune?

How do you decide what type of transfer learning you should perform on a new dataset? This is a function of several factors, but the two most important ones are the size of the new dataset, and its similarity to the original dataset.

Scenarios:

- New dataset is large and very different from the original dataset.

  Train from scratch the whole network.
- New dataset is large and similar to the original dataset.

  Fine-tune the whole network.

When and how to fine-tune?

How do you decide what type of transfer learning you should perform on a new dataset? This is a function of several factors, but the two most important ones are the size of the new dataset, and its similarity to the original dataset.

Scenarios:

- New dataset is large and very different from the original dataset.
  Train from scratch the whole network.
- New dataset is large and similar to the original dataset.
  Fine-tune the whole network.
- New dataset is small but very different from the original dataset.

When and how to fine-tune?

How do you decide what type of transfer learning you should perform on a new dataset? This is a function of several factors, but the two most important ones are the size of the new dataset, and its similarity to the original dataset.

Scenarios:

- New dataset is large and very different from the original dataset.
  Train from scratch the whole network.
- New dataset is large and similar to the original dataset.
  Fine-tune the whole network.
- New dataset is small but very different from the original dataset.
  Use SVM without transfer learning.

When and how to fine-tune?

How do you decide what type of transfer learning you should perform on a new dataset? This is a function of several factors, but the two most important ones are the size of the new dataset, and its similarity to the original dataset.

Scenarios:

- New dataset is large and very different from the original dataset.
  Train from scratch the whole network.
- New dataset is large and similar to the original dataset.
  Fine-tune the whole network.
- New dataset is small but very different from the original dataset.
  Use SVM without transfer learning.
- New dataset is small and similar to original dataset.

When and how to fine-tune?

How do you decide what type of transfer learning you should perform on a new dataset? This is a function of several factors, but the two most important ones are the size of the new dataset, and its similarity to the original dataset.

Scenarios:

- New dataset is large and very different from the original dataset.
  Train from scratch the whole network.
- New dataset is large and similar to the original dataset.
  Fine-tune the whole network.
- New dataset is small but very different from the original dataset.
  Use SVM without transfer learning.
- New dataset is small and similar to original dataset.
  Fine-tune only some higher-level portion of the network.

Transfer learning in image specific tasks

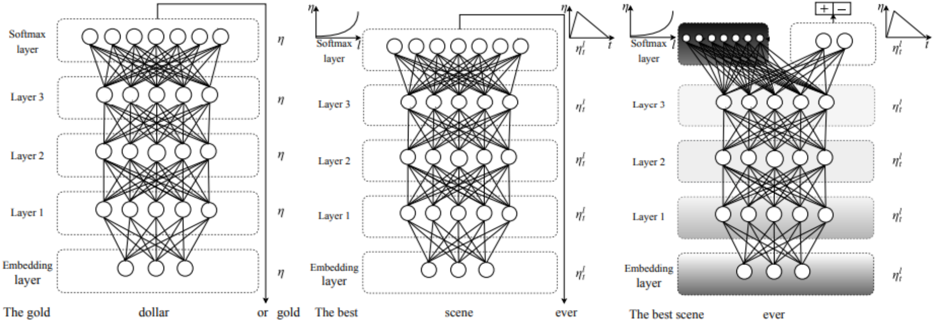
Transfer learning in NLP

The first successful transfer learning in NLP was done in 2018 which surpassed all text classification state-of-the-art. There was created framework ULMFiT, for transfer learning tasks in NLP.

Transfer learning in NLP

The first successful transfer learning in NLP was done in 2018 which surpassed all text classification state-of-the-art. There was created framework ULMFiT, for transfer learning tasks in NLP.

ULMFiT



(a) LM pre-training (b) LM fine-tuning (c) Classifier fine-tuning

Main ideas in ULMFiT

- Discriminative fine tuning

Main ideas in ULMFiT

- Discriminative fine tuning

  Different learning rates are used for different layers during the fine-tuning phase of LM (on the target task). This is done because the layers capture different types of information.

Main ideas in ULMFiT

- Discriminative fine tuning

  Different learning rates are used for different layers during the fine-tuning phase of LM (on the target task). This is done because the layers capture different types of information.

- Slanted triangular learning rates (STLR)

Main ideas in ULMFiT

- Discriminative fine tuning

  Different learning rates are used for different layers during the fine-tuning phase of LM (on the target task). This is done because the layers capture different types of information.

- Slanted triangular learning rates (STLR)

  Learning rates are first increased linearly, and then decreased gradually after a cut, i.e., there is a "short increase" and a "long decay."

Main ideas in ULMFiT

- Discriminative fine tuning

  Different learning rates are used for different layers during the fine-tuning phase of LM (on the target task). This is done because the layers capture different types of information.

- Slanted triangular learning rates (STLR)

  Learning rates are first increased linearly, and then decreased gradually after a cut, i.e., there is a "short increase" and a "long decay."

- Gradual unfreezing

Main ideas in ULMFiT

- Discriminative fine tuning

  Different learning rates are used for different layers during the fine-tuning phase of LM (on the target task). This is done because the layers capture different types of information.

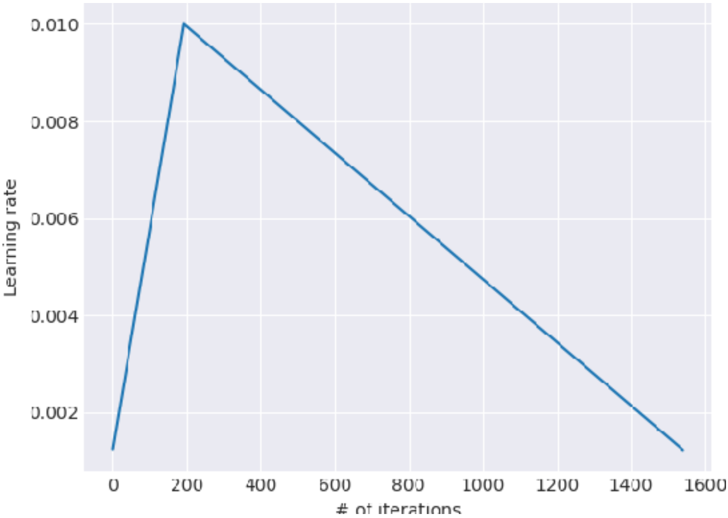
- Slanted triangular learning rates (STLR)

  Learning rates are first increased linearly, and then decreased gradually after a cut, i.e., there is a "short increase" and a "long decay."

- Gradual unfreezing

  During the classification training, the LM model is gradually unfreezed starting from the last layer.

STLR

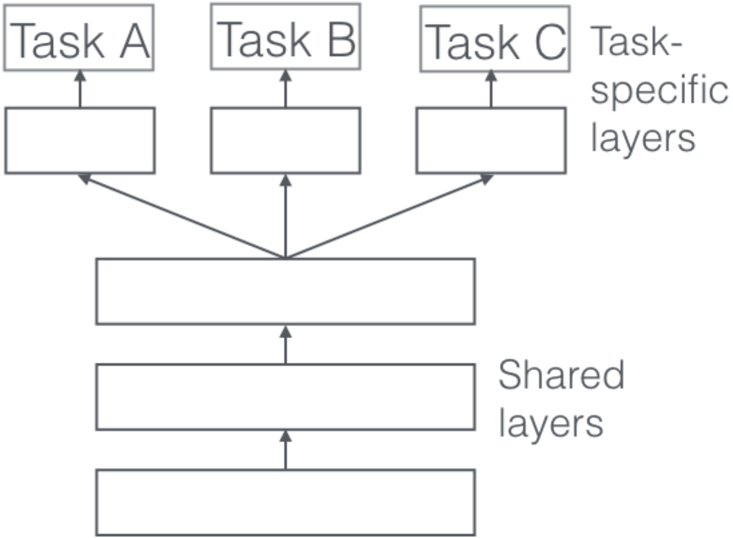


0.010
0.008
0.006
0.004
0.002
Learning rate
0
200
400
600
800
1000
1200
1400
1600
# of iterations

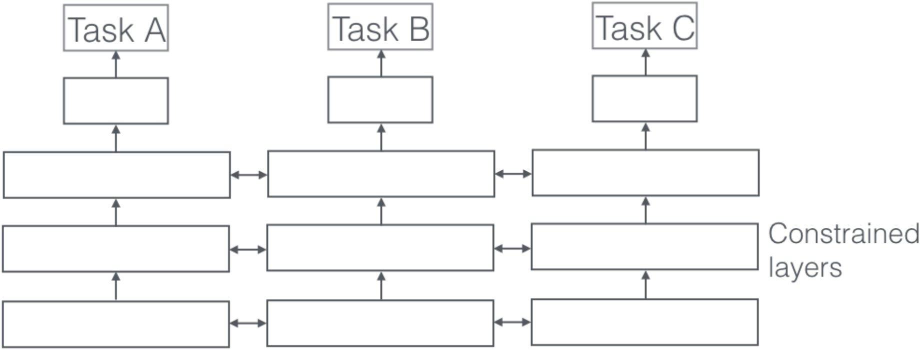
Outline

Multi-Task Learning

Multi-Task Learning



Multi-Task Learning



When does multi-task learning make sense?

When does multi-task learning make sense?

- Training on a set of tasks that could benefit from having shared lower level features.

Multi-Task Learning

When does multi-task learning make sense?

- Training on a set of tasks that could benefit from having shared lower level features.
- Amount of data you have for each task is quite similar.

Multi-Task Learning

When does multi-task learning make sense?

- Training on a set of tasks that could benefit from having shared lower level features.
- Amount of data you have for each task is quite similar.
- Can train a big enough neural network to do well on all the tasks.