Advanced Deep Learning and Kernel Methods

An Empirical Analysis of Transfer Learning and Implicit Regularization

Gabriele Pintus

January 13, 2025

University of Trieste

The problem

The problem

We aim to approximate a function

$$f : \mathbb{R}^d \to \mathbb{R}$$

using a neural network

$$h : \mathbb{R}^d \to \mathbb{R},$$

with parameters $\theta \in \Theta$.

The problem

We aim to approximate a function

$$f : \mathbb{R}^d \to \mathbb{R}$$

using a neural network

$$h : \mathbb{R}^d \to \mathbb{R},$$

with parameters $\theta \in \Theta$.

The challenge arises when the data generating distribution $\mathcal{D}$ is **difficult to sample from**, resulting in a small dataset

$$D = \{(x_i, y_i)\}_{i=1}^n$$

with limited samples.

The solution

Solving this problem with transfer learning means pre-training the network on another dataset, drawn from an easy-to-sample distribution, and then train it again a little bit on the small one (target dataset).
This last training phase is called **fine tuning**.

The solution

Solving this problem with transfer learning means pre-training the network on another dataset, drawn from an easy-to-sample distribution, and then train it again a little bit on the small one (target dataset).
This last training phase is called **fine tuning**.

Okay but... why should this work?

Transferability of group invariances

From the simple mathematicaly equation

$$\langle gx, t \rangle = \langle x, g^{-1}t \rangle, \forall x, t \in \mathcal{X}$$

we can deduce that once the group orbit (or part of it) is seen during training, the acquired invariance transfers to out-of-sample data points.

However, transfer learning relies on some further assumptions …

Transfer Learning

Feature extraction

Neural networks better approximate compositional functions with hierarchical structures. In deep models, this ability is reflected in a bias towards learning progressively abstract representations of the input across layers, transitioning from fine-grained (microscopic) features to more general (macroscopic) patterns.

- The **microscopic** representation contains less information but is more generalized across different inputs, making it **less dependent on the specific dataset**.
- The **macroscopic** representation is richer in information but less generalized across inputs, making it **more dependent on the specific dataset**.

Implicit assumptions of TL

When doing Transfer Learning one implicitely make two assumptions

Implicit assumptions of TL

When doing Transfer Learning one implicitely make two assumptions

Assumption 1: The Network can de splitted into two parts:

1. **Semantic encoder**: maps the input from the representation space to the semantic space
2. **Task decoder**: maps the input from the semantic space to the task specific space

Implicit assumptions of TL

When doing Transfer Learning one implicitely make two assumptions

Assumption 1: The Network can de splitted into two parts:

1. **Semantic encoder**: maps the input from the representation space to the semantic space
2. **Task decoder**: maps the input from the semantic space to the task specific space

Assumption 2: The Semantic Encoder it task independent.

Implicit assumptions of TL

When doing Transfer Learning one implicitely make two assumptions

Assumption 1: The Network can de splitted into two parts:

1. **Semantic encoder**: maps the input from the representation space to the semantic space
2. **Task decoder**: maps the input from the semantic space to the task specific space

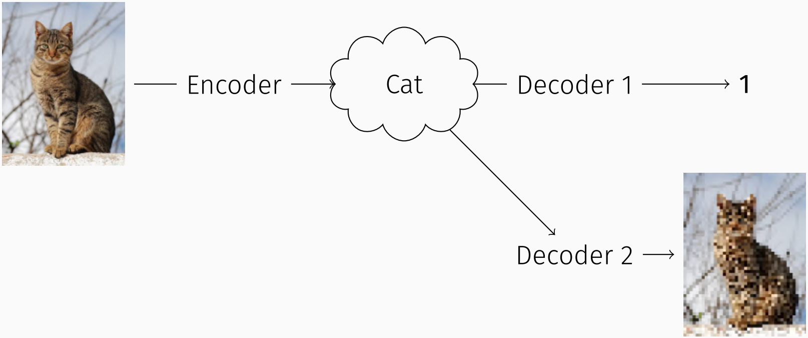
Assumption 2: The Semantic Encoder it task independent.

As a consequence, the encoder can be trained on any (similar) dataset, and we only need to train from scratch the decoder.

Example

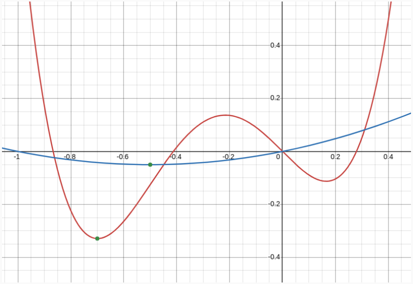
Example with two tasks:

1. **Classify** images as cats(1) or dogs(0).
2. **Compress** images of cats and dogs.



Regularization

Tranfer learning can also bee seen as a very good weights initialization strategy.
By using a more general pre-training dataset we can induce a regularization effect on the loss.



Intuitively, the pretraining phase restrict gradient descent to stay in a "good" region.

More formally

The paper "*Inductive biases of multi-task learning and finetuning: multiple regimes of feature reuse*" present an important result which explains how this fine-tuning approach introduces meaningfull biases regarding:

- Sparsity penalty
- Magnitude regularization
- Feature reuse

Proposition 3: Implicit Regularization Penalty

Fine-tuning Setup:

- Pretrain on auxiliary task: Learn $m_h^{\text{aux}}, \theta_h^{\text{aux}}, \gamma_h$.
- Fine-tune on main task: Update $m_h^{\text{main}}, \theta_h^{\text{main}}$, reinitialize γ_h.

Regularization Penalty:

$$R = \sum_{h=1}^{H} r(m_h^{\text{main}}, \theta_h^{\text{main}} | m_h^{\text{aux}}, \theta_h^{\text{aux}}),$$

with:

$$r = \left(\frac{m_h^{\text{main}}}{u^*} - \gamma_h \right)^2 + (u^*)^2 + m_h^{\text{aux}} - 2u^* \sqrt{m_h^{\text{aux}}} \langle \theta_h^{\text{main}}, \theta_h^{\text{aux}} \rangle.$$

u^* is the root of:

$$-m_h^{\text{main}} + \gamma_h u - m_h^{\text{aux}} \langle \theta_h^{\text{main}}, \theta_h^{\text{aux}} \rangle u^3 + u^4 = 0.$$

Insights from the Regularization Penalty

Key Terms in r:

- $\left(\frac{m_h^{\text{main}}}{u^*} - \gamma_h\right)^2$: Penalizes deviations in magnitude from initialization.
- $(u^*)^2$: Penalizes large updates during fine-tuning.
- $-2u^* \sqrt{m_h^{\text{aux}}} \langle \theta_h^{\text{main}}, \theta_h^{\text{aux}} \rangle$: Encourages alignment with pretrained directions.
- m_h^{aux}: Rewards reuse of pretrained features.

Emerging Behavior:

- Encourages feature reuse: Alignment term $\langle \theta_h^{\text{main}}, \theta_h^{\text{aux}} \rangle$.
- Promotes sparsity: Penalizes large changes in m_h^{main}.
- Balances task-specific adaptations with reliance on pretrained features.

Practical Implications

Impact on Transfer Learning:

- Fine-tuning prefers features aligned with pretraining directions.
- Sparse fine-tuning improves generalization on downstream tasks.
- Tasks with high feature overlap benefit most from fine-tuning.

Optimization Strategy:

- Rescale pretrained weights (γ_h) to control the tradeoff between feature reuse and adaptation.
- Prioritize tasks with correlated auxiliary and main task features.

The Experiment

The Experiment

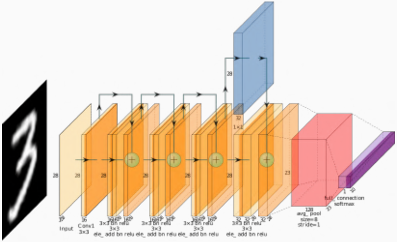
We want to test in practice the concepts we have seen so far.
In order to do so we fix:

- **Model architecture**: ResNet 50
- **Pre-training dataset**: ImageNet-1K
- **Fine tuning dataset**: MNIST

The Model

`Resnet-50` Introduced in 2015 by Kaiming He et al, $|\Theta| \approx 23.5 \cdot 10^6$

- 1 Initial Convolutional Layer
- 4 Stages of Residual Blocks
- Global Average Pooling Layer
- Fully Connected Layer with Softmax



Pre-training dataset

ImageNet-1K is a widely used subset of the larger ImageNet dataset. It contains 1.2 million images belonging to 1000 different classes.

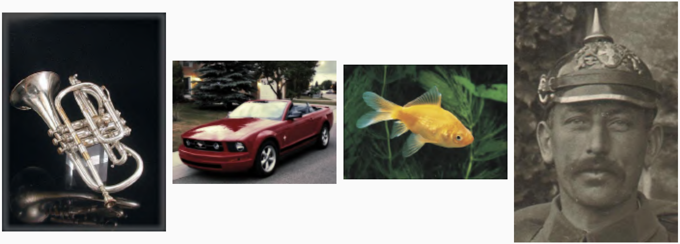


Figure 1: Four samples from ImageNet-1K.

Fine Tuning dataset

MNIST consists of 70K grayscale images of handwritten digits $[0, 9]$

Figure 2: Samples from MNIST.

ImageNet does not include digit images, meaning the pre-trained model has never explicitly encountered digits during its training.

To be consistent with the problem definition we downsample the dataset to just 50 data points.

What to measure

Aside from the usual metrics, loss and accuracy, we decide to track:

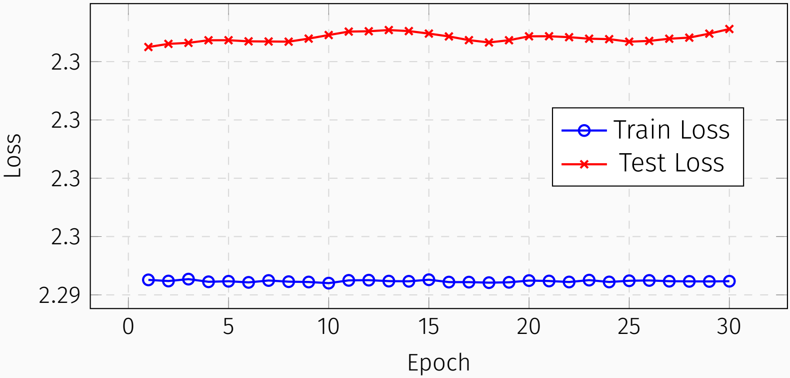
- gradient norm
- weights norm change
- weights alignment change

The last two computed both network-wise and layer-wise.

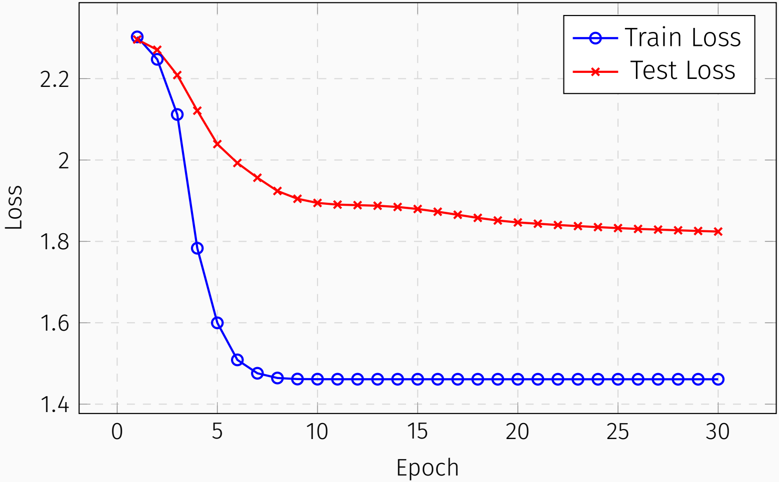
Best Loss

Model	Best Loss	Best Balanced Accuracy
New model	2.3025	11.05 %
Pretrained model	1.8348	61.69 %

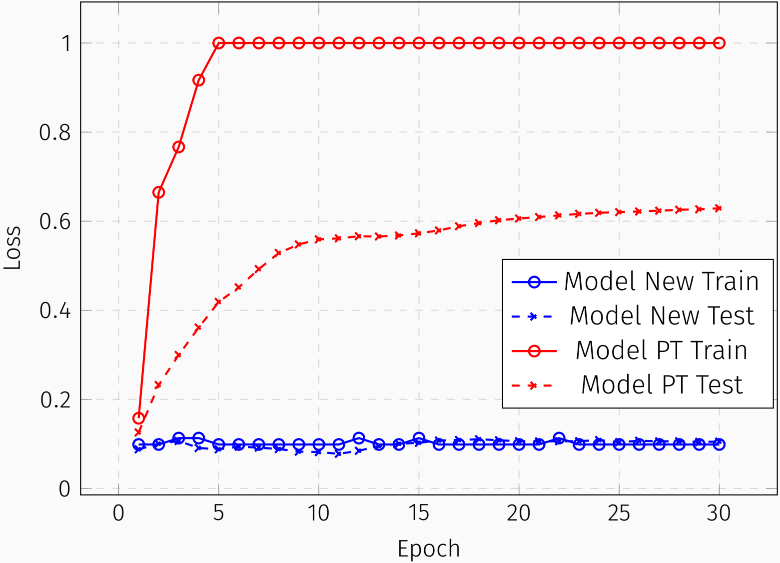
Loss Evolution - New model



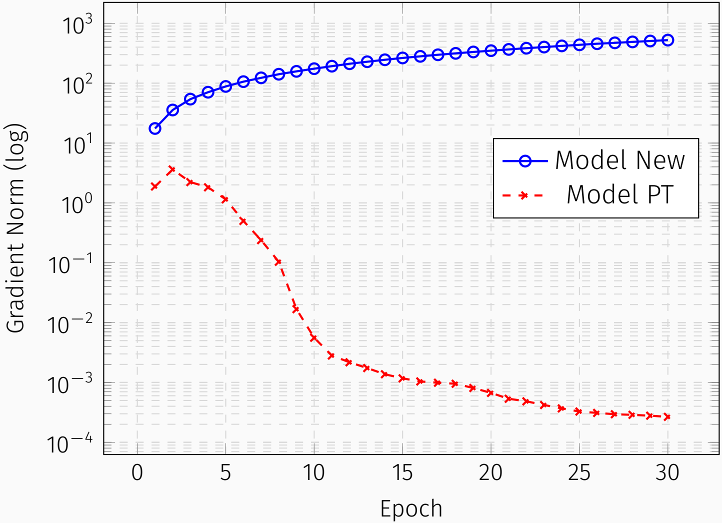
Loss Evolution - Pre-Trained model



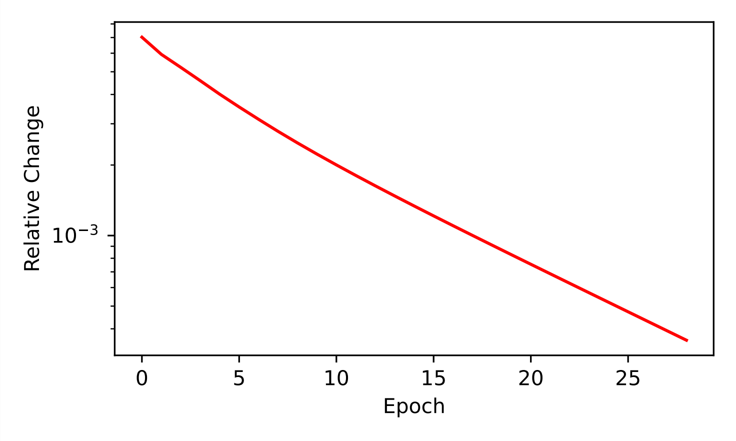
Accuracy Evolution



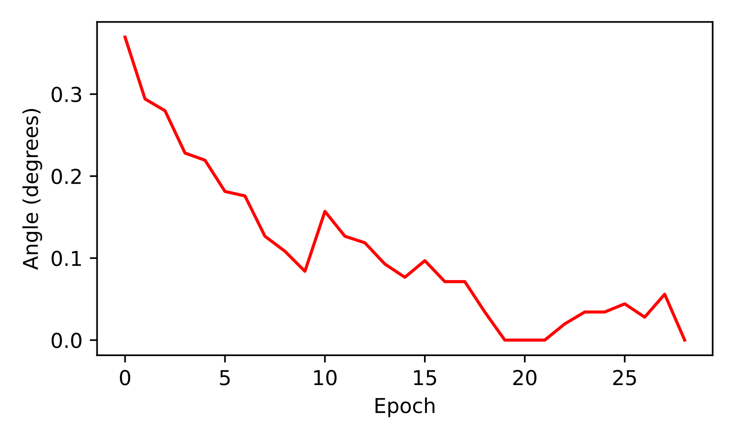
Gradient norm



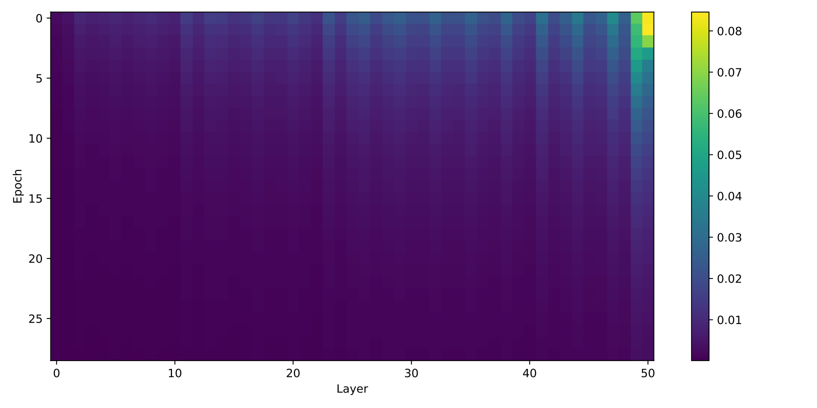
Overall magnitude change



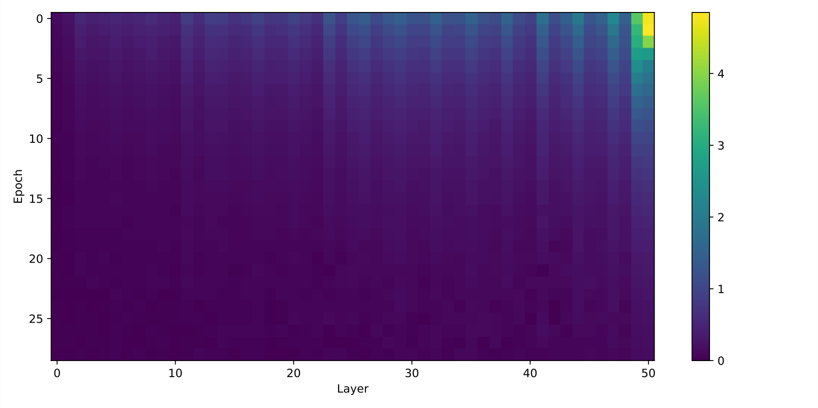
Overall angle change



Layerwise magnitude change



Layerwise angle change



Loss landscape

The Pre-Trained model has a clearly smoother loss.

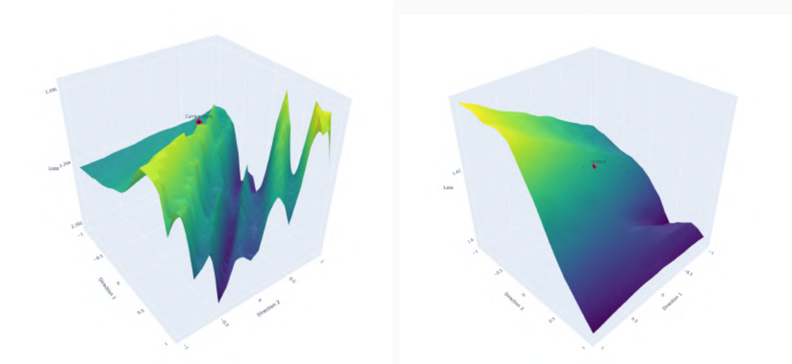


Figure 3: New model vs Pre-Trained model

Review of the assumptions

These last two plots clearly showed how the assumptions we made are (mostly) satisfied. Finally we can state that:

- The convolutional layers constitutes the semantic encoder
- The fully connected layer constitutes the task decoder
- Transfer learning help regularize the loss landscape

The End

Thank You!