Quantification of the transferability of features between deep neural networks

Romain Orhand, Hiba Khodji, Amarin Hutt, Anne Jeannin-Girardon

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2021) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

25th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems

Quantification of the transferability of features between deep neural networks

Romain Orhand[a], Hiba Khodji[a], Amarin Hutt[a], Anne Jeannin-Girardon[a]

[a]*University of Strasbourg, ICube Laboratory UMR 7357, 300 bd Sébastien Brant - CS 10413 - F-67412 Illkirch Cedex*

Abstract

The computationally expensive nature of Deep Neural Networks, along with their significant hunger for labeled data, can impair the overall performance of these models. Among other techniques, this challenge can be tackled by Transfer Learning, which consists in re-using the knowledge previously learned by a model: this method is widely used and has proven effective in enhancing the performance of models in low resources contexts. However, there are relatively few contributions regarding the actual transferability of features in a deep learning model. This paper presents QUANTA (QUANtitative TrAnsferability), a method for quantifying the transferability of features of a given model. A QUANTA is a 2-parameters layer added in a target model at the level at which one wants to study the transferability of the corresponding layer in a source model. Data from the target domain being fed to both the source and the target models, the parameters of the QUANTA layer are trained in such a way that a mutually exclusive quantification occurs between the source model (trained and frozen) and the (trainable) target model. The proposed approach is evaluated on a set of experiments on a visual recognition task using Convolutional Neural Networks. The results show that QUANTA is a promising tool for quantifying the transferability of features of a source model, as well as a new way of assessing the quality of a transfer.

Keywords: transfer learning; feature transferability quantification; convolutional neural networks; deep learning

1. Introduction

Deep Neural Networks are praised for their ability to extract hierarchical features from data and used for solving various recognition tasks such as image recognition [10, 8], speech recognition [4], natural language processing [19], bioinformatics [16], etc. Such models are hungry for resources, either computation-wise or data-wise. Lack of computing resources induces an important training time for a model. Training time can be reduced by reducing the number

* Anne Jeannin-Girardon
E-mail address: anne.jeannin@unistra.fr

of trainable parameters, possibly at the cost of no longer being able to capture subtle underlying complexity within the data. Obviously, acquiring more computing power also alleviates training time. Lack of data can be addressed by augmenting the training dataset [11] or obtaining more training data. In both cases, and in some application domains, neither options are possible: an example is the context of studying rare disease genotypes for which, by definition, only few data are available. Either way, resource management is a major stake in machine learning and lacking either of these resources causes the performance of the model to drop.

An alternative answer to the problem of resource management is to re-use the knowledge previously learned by a model into another model, a method commonly known as *transfer learning*. Transfer learning is a machine learning paradigm focusing on applying knowledge gained from a source model to an untrained target model in order to (1) reduce training time or (2) minimize the amount of data required for training, and (3) improve the performance of a model on a target task. If objectives (1) or (2) and objective (3) are achieved, the transfer is considered a success. If the performance of a model, post-transfer, is lower than its performance without transfer, then the transfer is deemed unsuccessful: this phenomenon is known as negative transfer, and is usually caused by a dissimilarity between the source and the target domains that can negatively impact the performance on the target task [17]. A promising direction for avoiding this issue and to effectively apply transfer learning is to address the following questions [18]:

- "What to transfer": is it possible to determine which features or parameters should be transferred from a trained source model in order to improve the performance of a model on a target task?
- "When to transfer": the similarity of the source and the target domains should be assessed in order to avoid a negative transfer.
- "How to transfer": what actual methods should be used to carry out the transfer.

Given the availability of data and the (dis)similarity between both source and target domains and tasks, transfer learning problems can be categorized into three groups [2, 15, 14]: Inductive Transfer Learning, where the target task is different-but-related-to the source task and the source and target domains may or may not be the same; Transductive Transfer Learning, where the target domain is different-but-related-to the source domain and the target task to learn is the same as the source task. Such transfer applies when the feature spaces between the source and target domains are different or when they are similar but the marginal probability distributions are different (Domain Adaptation). Lastly, Unsupervised Transfer Learning is an instance of Inductive Transfer Learning where no labelled data are provided in either the source or the target domain.

Within the context of *Homogeneous* Transfer Learning (that is, when the feature spaces of the source and the target domain are the same), different approaches can be considered in order to address the "what to transfer?" question [14]: training instances from the source domain can be weighted to use only training data having a positive influence in the target domain; new representations can be learned in order to fill the gap between the source and the target domains; the commonalities between the source and the target domains can be learned and used to learn to solve the target task. The present study focuses on Parameter-based transfer: this approach, probably the most widely used, assumes that the parameters learned by the source model can be transferred to the target model. This process occurs at the layer level and the transferred parameters can either be frozen (*i.e.* un-altered during the target model training process) or fine-tuned (*i.e.* adjusted during the target model training process).

In the context of a *Homogeneous*, *Inductive* and *Parameter-based* transfer, this paper introduces QUANTA (QUANtitative TrAnsferability), a tool to evaluate the transferability of models at the layer level. The primary purpose of this tool is not to provide a methodology for transferring knowledge *per se*, but rather an analytical approach that would allow to further address the questions "what" and "when" to transfer.

This paper is organized as follows: the next section presents related works focusing on methodology around transfer learning. QUANTA are introduced in section 3, before experimental results are detailed and discussed in section 4. Finally, conclusions and future research directions are highlighted in section 5.

2. Related works

In the following, the works described were conducted in the context of Homogeneous, Inductive and Parameter-based transfer learning for visual recognition tasks.

The seminal paper [18], that promoted the study of feature transferability in deep neural network, aimed at addressing the question of which learned features can be considered general (and thus, transferable), or specific to a given domain and task. The transferability of features is measured as the accuracy of the target model after the transfer of the first n layers. Both source and target domains are the same ($\mathcal{D}_S = \mathcal{D}_T$), and two sorts of transfer were done: one where both source and target tasks are the same ($\mathcal{T}_S = \mathcal{T}_T$) and one in which the target task differs from the source task ($\mathcal{T}_S \neq \mathcal{T}_T$). Several observations were made:

- Parameters from low-level layers are general and transferable.
- Transferring higher-level layers impairs the performance of the target model. These layers are more task-specific.
- Fine-tuning the transferred parameters results in better performance, especially when the transferred parameters are task-specific.
- Features become less transferable as the distance between the source and target tasks increases.
- Transferring features, even from a model trained on an unrelated task, leads to better overall performance.
- The presence of learned co-adapted features in a model (that is, features which interact with each other over several layers and that cannot be re-learned by the upper layer alone) has a negative impact on transferability. Those features were highlighted by a performance drop followed by a recovery, within the network.

Most studies related to Transfer Learning aim to improve the performance of deep learning models, for instance by highlighting best practices in terms of model architecture, hyperparameters, etc., when applying transfer learning [1]. More specific works focus on data selection for training, for instance in the case of source-target joint training such as [5]: target training instances are selected when their low-level features are highly similar to those of the data from the source domain, for which abundant data is available. The pre-trained and the target models are jointly fine-tuned using only the selected data (this process is referred to as selective joint fine-tuning).

The work presented in [13] concerns the use of learned mid-level representation and how they can be applied for solving a target task different from the source task (*i.e.* whose label space is different from this of the source task). To do this, they transfer and freeze all mid-level representations (including those learned by the fully-connected layers usually found near the output of a Convolutional Neural Network) in the target model, and a fully-connected adaptation layer is added in order to adapt those representation to solve the target task. CactusNet [3] is an approach developed in order to apply learned features to a given piece of data, based on the hypothesis that a model has an individual response (or applicability) for each datapoint. The idea is to *grow* a pre-trained source model by creating branches from the most applicable features in order to specialize the model to the target task by learning higher-level features that are not yet known in the target data. This approach also has intrinsic applications regarding unsupervised learning and multi-task learning.

Cross-stitch networks [12] are used in the context of multi-task learning as well. These models learn linear combinations of shared and task-specific representations between multiple models using cross-stich units (CSU) that combine activations maps from the models and use these combinations as inputs for the next layer of each model. Parameters Transfer Units (PTU) [20] also apply combinations between activations maps, and those combinations are non-linear. Data from the target domain are fed to both the (frozen) source and the target model and the combinations learned by the PTU are fed only to the target network. The overall specification of a PTU exhibits more complexity than that of a CSU because a PTU is made of two distinct gates, used to (1) adapt source activations to the target domain and (2) combine target activations with the adapted source activations. It turns out that not only PTU are used for transferring parameters from the source to the target model, but their gates also provide some insights regarding the actual transferability of these parameters (*i.e.* to which extent parameters learned by the source model can be re-used by the target model): while this transferability is usually expressed in terms of the target model accuracy, PTU provides an "absolute" value that is not expressed as a function of the performance of the target model.

QUANTA is conceptually close to both CSU and PTU in the sense that it performs combinations of elements from the source and the target model. Beside the fact that the primary purpose of QUANTA is to evaluate the transferability of layers (rather than actually perform a transfer), it differs from CSU et PTU in three ways:

1. QUANTA does not combine activations maps but the weighted inputs of layers.
2. This combination is less complex and it involves adding only two more parameters in the model.

3. Most importantly, the combination between the source and the target models is mutually exclusive (*i.e.* during training, a *choice* has to be made between the two models) and the expression of this combination provides a metrics that is not a function of the performance of the target model.

These three points are detailed in the next section.

3. Quantitative measure of layer transferability with QUANTA

Before diving into the technical specification of QUANTA, it is useful, first, to detail the intuitions and ideas behind this proposition.

A source model S is trained on data drawn from $\mathcal{D}$ to solve a classification task $\mathcal{T}_S$. The objective is to train a model T on data also drawn from $\mathcal{D}$, to solve a classification task $\mathcal{T}_T$ ($\mathcal{T}_T$ may or may not be the same as $\mathcal{T}_S$), while re-using the features learned by S as much as possible (but not at the expense of the performance of T).

Assessing to which extent the features of S can be used for solving $\mathcal{T}_T$ comes down to feeding the data drawn for T to S, use an output layer adapted to task $\mathcal{T}_T$ to compute the predictions, and check the accuracy of S: this assessment is basically an inductive paramater-based transfer of all the hidden layers of S.

In order to assess how a particular hidden layer n of S can be re-used for solving $\mathcal{T}_T$, both models S and T must be considered at the same time: models S and T have the same architecture, except for the output layer, which is specific to the task each model must solve; the parameters of T are initialized using a legacy initialization scheme such as Xavier's and they are trainable. At no point in the process are the parameters of the first n layers of S copied into T: S parameters are frozen and, while training T, the training instances are fed simultaneously to S and T.

In T, the activation of layer n is computed as a function of the weighted inputs of layers n of S and T: $A_{n,T} = f(Z_{n,S}, Z_{n,T})$ and the signal keeps propagating until T outputs a prediction: this process is depicted on figure 1.

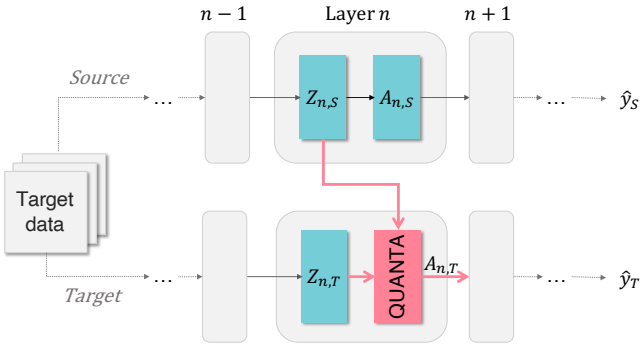


Fig. 1: Implementation of a QUANTA layer in the target model T: the activation of the n^{th} layer in T is expressed as a weighted combination of the weighted inputs of layers n of both T and the source model S.

In the case of a "regular" inductive parameter-based transfer, the activation of layer n in T only uses $Z_{n,S}$. Otherwise, $Z_{n,S}$ and $Z_{n,T}$ can be weighted: if $Z_{n,S}$ has an important weight and the prediction of T is wrong, the weight of $Z_{n,S}$ must decrease. At the same time, the parameters of T are also adjusted and, if the classification error of T is reduced, their weight increases. Consequently, T has two more parameters to learn: the weights $v_{n,S}$ and $v_{n,T}$ respectively applied to $Z_{n,S}$ and $Z_{n,T}$. These two parameters express to which extent the parameters of S can be re-used or if T should use its own (learned) parameters.

Both parameters are implemented in a layer (a QUANTA layer) at the n^{th} layer within the target model T in order to compute $A_{n,T} = f(Z_{n,S}, Z_{n,T})$. Defining:

$$\Lambda_n = \begin{pmatrix} \lambda_{n,S} \\ \lambda_{n,T} \end{pmatrix} = \text{softmax} \begin{pmatrix} v_{n,S} \\ v_{n,T} \end{pmatrix},$$

the activation of layer n of the target model is now expressed as:

$$A_{n,T} = f(\lambda_{n,S} Z_{n,S} + \lambda_{n,T} Z_{n,T}), \tag{1}$$

where $f()$ is an activation function. The weights $v_{n,\cdot}$ are updated using the prediction error C_T of T: $v_{n,\cdot} = v_{n,\cdot} - \eta_v \partial C_T / \partial v_{n,\cdot}$.

Being computed using a *softmax* function, the scalars $\lambda_{n,S}$ and $\lambda_{n,T}$ range within $[0; 1]$ and $\lambda_{n,S} + \lambda_{n,T} = 1$. Considering the source model, the scalar $\lambda_{n,S}$ denotes the degree to which the source parameters can be used as-is so that the target model can solve its task. Thus, this value indicates how much the n^{th} layer of the source model is transferable: a value close to 1 (resp. 0) means that the source parameters are transferable (resp. not transferable). The scalar $\lambda_{n,T}$, associated to the target model, acts as a learning adjustment inversely proportional to the transferability $\lambda_{n,S}$ and it provides information about the degree of modification that should be applied to the transferred parameters. Using the softmax function to compute Λ_n makes S and T mutually exclusive: if the features of S are not adapted then T must compensate by learning other features, and the parameters $v_{n,\cdot}$ change according to *how much of a layer n from the source model S can be expressed through the activated output $A_{n,T}$ of the target model T*.

When applying transfer learning, the ideal situation is to *maximize* the use of the features of S to solve a target task $\mathcal{T}_T$. However, in the context of an inductive parameter-based transfer, the parameters of a set of layers are merely copied into the target model and possibly fine-tuned to adapt them to $\mathcal{T}_T$. Could the actual transferability of features, layer-wise, be evaluated? Or, to put it another way, is it possible to evaluate the quantity of features that can be re-used to solve a different task? QUANTA is an attempt at such an evaluation, through the combination computed between S and T. However, given the intrinsic complexity of deep neural networks, it would be excessive to claim that a QUANTA layer *maximizes* the use of features of S for solving $\mathcal{T}_T$.

That being said, in order to shift the system towards *prioritizing* the use of these features over the features that T would learn from scratch, the initial value of $v_{n,S}$ and $v_{n,T}$ are set such that $\lambda_{n,S} \approx 0.9$ and $\lambda_{n,T} \approx 0.1$. Since the fundamental idea is to reuse the features learned by S, *the target model should be prevented from learning,* de novo*, features for solving $\mathcal{T}_T$ if these are already known by the source model* (and thus, transferable). The proposed initialization is expected to have the following effect: if, initially, compelling T to use $Z_{n,S}$ gives rise to a significant error on a given input from the target dataset, $\lambda_{n,S}$ decreases because its associated error gradient is more important (in other words, the features that T attempted to use were not relevant for solving $\mathcal{T}_T$). Since $\lambda_{n,S}$ and $\lambda_{n,T}$ are bound by a softmax, $\lambda_{n,T}$ will increase proportionally, meaning that T adjusts the features learned by S. Initially maximizing the use of $Z_{n,S}$ prioritizes S over T and, as the training process goes on, T compensates for unadapted features by adjusting them.

The influence of the initial values of the QUANTA layer parameters is discussed in the next section, in which experimental results are presented after the protocol for running these experiments is described.

4. Experimental results

The experiments presented thereafter aim at providing inputs to the following questions:
(1) Does the inclusion of a QUANTA layer in a model impair the learning process of the model?
(2) Does the parameter $\lambda_{n,S}$ provide some information regarding the transferability of the layer n, from the source model, to solve $\mathcal{T}_T$?
(3) What is the influence of the initial value of Λ_n?
(4) How can $\lambda_{n,S}$ actually be interpreted?

Before discussing these questions, the experimental protocol is presented in the next section.

4.1. Experimental protocol

The data used to conduct the experiments with QUANTA are the CIFAR-10 and the CIFAR-100 datasets [9] (both are built from the Tiny Image dataset). CIFAR-10 contains 60 000 32×32 color images split into 10 classes of 6000 images each. The dataset contains 50 000 training images and 10 000 testing images. CIFAR-100 is very similar to

CIFAR-10, except that the images are split into 100 classes of 600 images each. Each class has 500 training images and 100 testing images. The labels of CIFAR-10 and CIFAR-100 do not overlap, so the tasks associated to each dataset are different (the label spaces are different).

CIFAR-10 is the dataset used to train the source model (source task `A` = $\mathcal{T}_S$). `B` is the target task corresponding to the CIFAR-100 dataset. To test the QUANTA system, two sets of experiments have been defined: experiments `AnB` (resp. `AnA`) refer to the measure of the transferability of layer n in the source model when the target model attempts to solve task `B` (resp. `A`). Experiments `AnA` can be seen as control experiments, since it is expected that re-using the features learned by S to solve the same task it was trained on would lead to a successfully trained target model.

The architecture of the convolutional neural networks used is depicted on figure 2: both the source and the target model have the same architecture (except for the output layer, adapted to the task the target model should solve) made of three convolutional blocks (each block contains two convolutional layers) computing, respectively, 64, 256 and 512 3×3 feature maps. A fully connected layer with 2048 units serves as a classifier and a final layer computes the output for the task at hand. This last layer is not meant to be transferred, which means that there are 7 layers, in the source model, whose transferability can be assessed (the blue layers on figure 2).

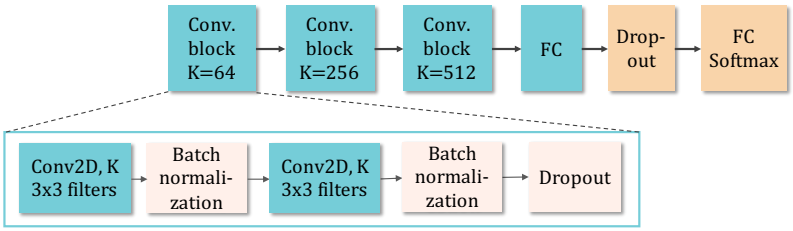


Fig. 2: Convolutional Neural Network architecture used for the source and the target models. The layers depicted in blue are the layers whose transferability can be assessed.

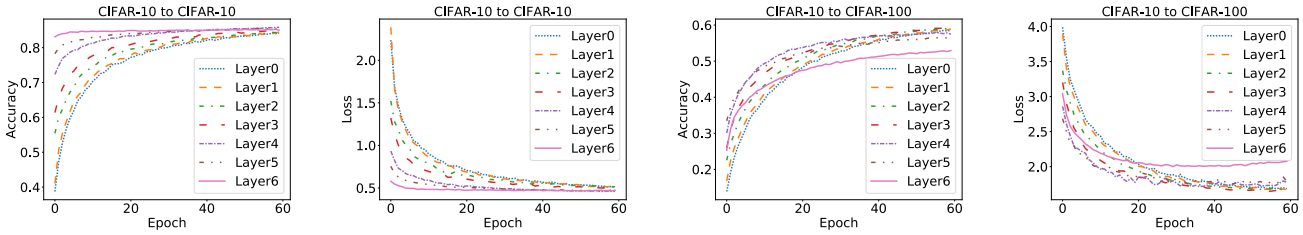
It has been empirically observed that complex Convolutional Neural Networks are prone to learning co-adapted features [7, 6], so a larger than necessary model has been used to test the ability of QUANTA to highlight learned co-adapted features.

The source model was trained during 60 epochs on CIFAR-10 using 32-instances batches. Training data was augmented with random rotations, pixel shifts, image crops, zooms and symmetries. The applied learning rate was 10^{-4}. The training process was optimized with the Adam optimizer (weight decay: 10^{-6}) using a cross-entropy categorical loss. This configuration is the same for training the target model T.

An experiment is defined as the assessment of the transferability of layer n in the source model when the target model T attempts to solve task `A` or `B`. For a given target task, there was thus 7 experiments (that is, one experiment per transferable layer). Each experiment was repeated 30 times. During each run, and at the end of each training epoch, T is evaluated on the testing data and the following metrics are collected: the transferability $\lambda_{n,S}$ of layer n, the top-1-accuracy of the target model as well as its loss, its precision and recall and its top-3-accuracy. For a given run, the final metrics are computed as the average of the last 10 recorded metrics (in order to take into account the stochastic variability of the evaluation from one epoch to another). Then, the final metrics are averaged over the 30 runs of the experiment. Experiments were run for three different initial values of $\lambda_{n,S}$: 0.90, 0.75 and 0.50.

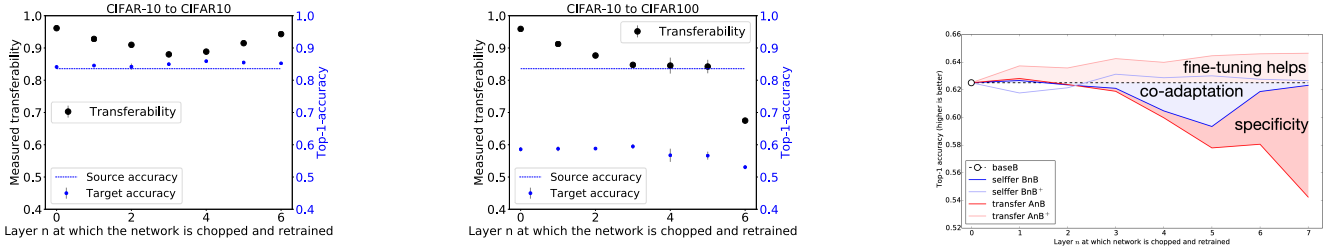
A common way to check on the learning process of a model is to examine its loss and top-1-accuracy during training. Figure 3 depicts the evolution of those metrics, for $\lambda_{n,S}^{\text{init}} = 0.90$: for both task, the accuracy increases while the loss decreases, thus confirming that integrating a QUANTA layer within the target model did not impair the training process of this latter.

The results presented in the rest of this section are graphs depicting the measured transferability per layer (black dots) along with the top-1-accuracy associated to the layer (blue crosses –the dashed line is the top-1-accuracy of the source model on task `A`). Firstly, the overall dynamics of the measured transferability with $\lambda_{n,S}^{\text{init}} = 0.90$ is presented (section 4.2). More experiments are detailed in section 4.3 to try and characterize the influence of $\lambda_{n,S}^{\text{init}}$.



(a) Task `AnA` (b) Task `AnA` (c) Task `AnB` (d) Task `AnB`

Fig. 3: Accuracy (a, c) and loss (b, d) of the target model during training; $\lambda_{n,S}^{\text{init}} = 0.9$



(a) Task `AnA` (b) Task `AnB` (c) Reference results from [18]

Fig. 4: Dynamics of the measured transferability for tasks `AnA` and `AnB` with $\lambda_{n,S}^{\text{init}} = 0.90$

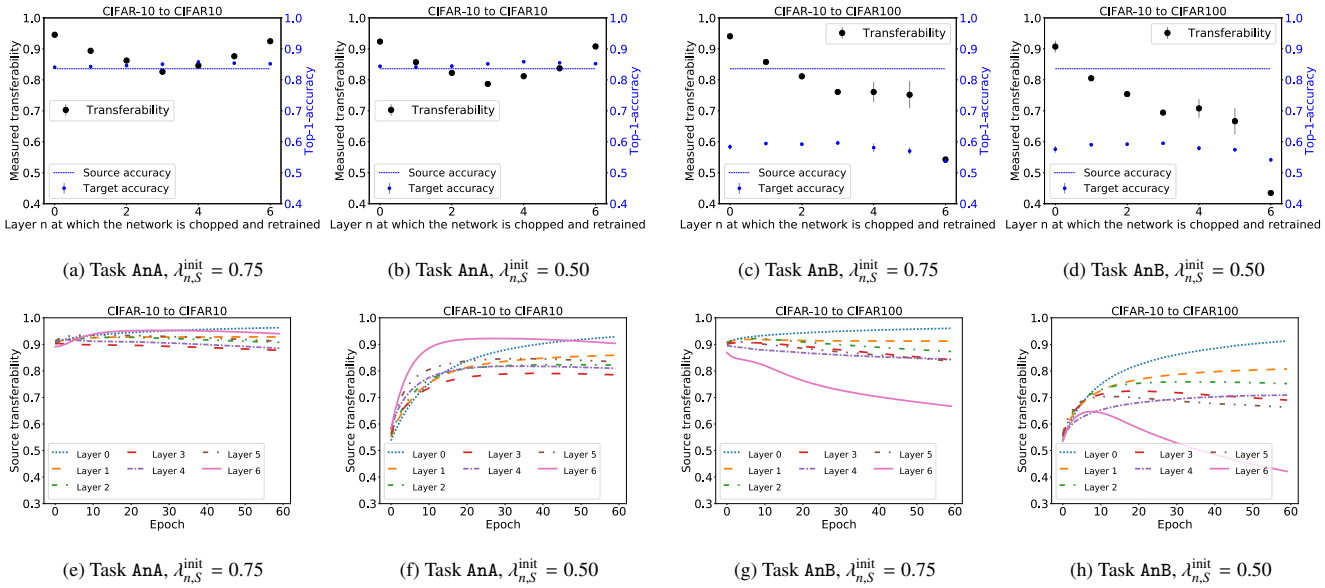
4.2. Dynamics of the measured transferability

Figure 4 presents the dynamics of the measured transferability (fig. 4a and 4b) for tasks `AnA` and `AnB`, compared with the seminal results of [18] (fig. 4c) in which the transferability is expressed as the top-1-accuracy of the target model after transferring the first n layers. The transferability as expressed by QUANTA does not depend on a performance metric: this is why the *dynamics* of the transferability is looked at (that is, how it evolves according to the layer considered).

It should be noted that the experiments presented in [18] were conducted on the ImageNet dataset with a different CNN architecture. Since it is acknowledged that hierarchical features learned by CNN are more specific as the considered layer is deeper, it can be assumed that the dynamics showed in [18] can be generalized to most image datasets (PTU [20] exhibited this dynamics when evaluating a transfer between CIFAR-10 and CIFAR-100).

For task `AnA` (fig. 4a), the dynamics of QUANTA over the layers is the same as the dynamics of model `selfer BnB` of fig. 4c: a drop of transferability, followed by a recovery, is observed at layer 3. This can be interpreted as the learning of co-adapted features by the source model. Regarding task `AnB` (fig. 4b), a significant transferability drop occurs at level 6: this was expected since the tasks are different: the classifier at the end of the source model cannot be re-used as-is to solve task $\mathcal{T}_T$ = `B`. This dynamics is also the same as the dynamics of fig. 4c (`transfer AnB`).

In fig. 4c, `transfer AnA+` and `transfer AnB+` show the target accuracy when fine-tuning is applied to the transferred parameters. In both cases, the accuracy is maintained and constant. In the case of QUANTA (fig. 4a and 4b), the target accuracy is also maintained and constant: for task `AnA`, the target accuracy (0.8488 –averaged over all layers) is at the level of the source accuracy (0.84). This was expected since the training data for T and S are the same and $\mathcal{T}_S = \mathcal{T}_T$. Regarding task `AnB`, the average target accuracy over all layers is 0.5743 (and, layer-wise, this accuracy drops slightly as n gets closer to 6). This is far below the source accuracy but this was also expected since (1) the source network is fed the target training data to pass $Z_{n,S}$ to the target model and (2) QUANTA does not apply fine-tuning.



(a) Task AnA, $\lambda_{n,S}^{\text{init}} = 0.75$ (b) Task AnA, $\lambda_{n,S}^{\text{init}} = 0.50$ (c) Task AnB, $\lambda_{n,S}^{\text{init}} = 0.75$ (d) Task AnB, $\lambda_{n,S}^{\text{init}} = 0.50$

(e) Task AnA, $\lambda_{n,S}^{\text{init}} = 0.75$ (f) Task AnA, $\lambda_{n,S}^{\text{init}} = 0.50$ (g) Task AnB, $\lambda_{n,S}^{\text{init}} = 0.75$ (h) Task AnB, $\lambda_{n,S}^{\text{init}} = 0.50$

Fig. 5: Top-row: comparison of the final transferability for tasks AnA and AnB, layer-wise, for different values of $\lambda_{n,S}^{\text{init}}$. Bottom-row: evolution of $\lambda_{n,S}$ during training.

4.3. Influence of the QUANTA parameters initial values

The experiments presented hereafter were conducted with two more values of $\lambda_{n,S}^{\text{init}}$: 0.75 and 0.50. Figure 5, top row, shows the measured transferability and the target accuracy for these initial values. For task AnA (resp. AnB), regardless of the initial value of $\lambda_{n,S}^{\text{init}}$, the target accuracy is ≈0.84 (resp. ≈0.57); this holds true for the other collected metrics (data available in the supplementary materials). This observation suggests that it is indeed possible to prioritize (if not maximize) the use of the features learned by S to solve a target task and a lower measured transferability would indicate that the target learned more features from scratch.

The bottom row of fig. 5 illustrates the evolution of $\lambda_{n,S}$ during training. The following comments can be made: using $\lambda_{n,S}^{\text{init}} = 0.50$ (fig. (f) and (h)) means that initially, no model is favored over the other. In this case, and regardless of the task, $\lambda_{n,S}$ increases during at least 10 epochs: the system attempts to use the features learned by S instead of relying on the randomly initialized parameters of T. This observation is in line with the experiments of [18] suggesting that transferring parameters from a not so related task leads to a better top-1-accuracy than starting training from random parameters. It is interesting to note that, for task AnB, the system first reinforces the use of the parameters of layer 6 before giving up because this layer is too specific to the source task.

When $\lambda_{n,S}^{\text{init}} = 0.75$ (fig. (e) and (g)), the system is initially compelled to apply the features learned by S. However, $\lambda_{n,S}$ quickly decreases, which means that the prediction error originates mostly from the source and that reducing $\lambda_{n,S}$ reinforces the adjustment made by the target.

To sum up, the results presented above provide some insights regarding the questions raised at the beginning of this section. Firstly, including a QUANTA layer in a model, whichever the layer, do not impair the learning process of the model: this latter minimizes its cost function and metrics such as the top-1-accuracy, the precision, the recall, etc. improve during training.

The dynamics exhibited by $\lambda_{n,S}$ over the n layers of the source model is coherent with the degree of genericity or specificity of the layer, as observed in previous studies such as [18, 20, 6]. Moreover, the results also highlighted the presence of learned co-adapted features. Those two points denote the ability of $\lambda_{n,S}$ to express a *degree of transferability* of a layer n.

However, this *degree of transferability* is not yet clear: indeed, while different initial values of $\lambda_{n,S}$ led to the same overall top-1-accuracy, the lower $\lambda_{n,S}^{\text{init}}$, the lower the final measured transferability. Nonetheless, this seems to indicate that a higher initial value of $\lambda_{n,S}^{\text{init}}$ allows prioritizing the use of features learned by S to solve a target task while

preventing T from learning these features from scratch, thanks to the mutually exclusive quantification computed by QUANTA.

Now, the question is to interpret the final value of $\lambda_{n,S}$ for a given layer. Assuming the computed $\lambda_{n,S}$ of layer $n = 0$ for task `AnB` is 0.97, it seems incoherent to state that $0.97 \times \text{layer}_n$ can be transferred while the top-1-accuracy of this layer in the target model for this task barely reaches 0.60: the "actual" transferability may actually be a function of both the measured $\lambda_{n,S}$ *and* the accuracy of T when transferring n. Fruther investigations are required in order to better frame the interpretation of this measured transferability and to figure out how this metric can be exploited in practice.

Further considerations about this work also include the fact that, as it is, it is constrained to neural network architectures that are the same for both tasks.

5. Conclusion

While transfer learning is a well-established method to deal with the lack of labelled data and/or the lack of computing resource, it is still tricky to anticipate which parameters inferred on a source task are re-usable to solve a (related) target task and to which extent these parameters should be adjusted in order to solve this target task. The system QUANTA (QUANtitative TrAnferability) introduced in this paper is an attempt at providing a quantitative measure of transferability of the parameters of a trained deep neural network. To evaluate the transferability of the first n layers of a source model to a target model, QUANTA combines the weighted inputs of both the source and target models to computes the target activation of the considered layer. This combination is mutually exclusive, so if a layer is not transferable, the system indicates what amount of adjustment is required to compensate the lack of transferability. QUANTA was tested on a visual recognition task using Convolutional Neural Networks and the datasets CIFAR-10 (source task) and CIFAR-100 (target task). Not only is the learning process of the target model unaffected by the inclusion of QUANTA, but the observed dynamics of the measured transferability is consistent with previous studies that measured the transferability of a source model as the accuracy of the target model post transfer. These encouraging results must now be reinforced by further study, firstly by conducting more experiments (using other data and tasks, possibly other than visual recognition tasks) in order to better frame the interpretation of this measured transferability. Other future works include studying the transferability of multiple source models or even to attempt tranferring knowledge learned on heterogenous data for solving a common task.

References

[1] Azizpour, H., Razavian, A.S., Sullivan, J., Maki, A., Carlsson, S., 2015. Factors of transferability for a generic convnet representation. IEEE transactions on pattern analysis and machine intelligence 38, 1790–1802. Publisher: IEEE.

[2] Chowdhury, S.B.R., Annervaz, K., Dukkipati, A., 2018. Instance-based Inductive Deep Transfer Learning by Cross-Dataset Querying with Locality Sensitive Hashing. arXiv preprint arXiv:1802.05934 .

[3] Collier, E., DiBiano, R., Mukhopadhyay, S., 2018. Cactusnets: Layer applicability as a metric for transfer learning, in: 2018 International Joint Conference on Neural Networks (IJCNN), IEEE. pp. 1–8.

[4] Deng, L., Hinton, G., Kingsbury, B., 2013. New types of deep neural network learning for speech recognition and related applications: an overview, in: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 8599–8603. doi:10.1109/ICASSP.2013.6639344.

[5] Ge, W., Yu, Y., 2017. Borrowing Treasures from the Wealthy: Deep Transfer Learning through Selective Joint Fine-Tuning, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, Honolulu, HI. pp. 10–19. URL: http://ieeexplore.ieee.org/document/8099492/, doi:10.1109/CVPR.2017.9.

[6] Gumbira, A., Kożuszek, R., 2018. Does fragile co-adaptation occur in small datasets?, in: 2018 Baltic URSI Symposium (URSI), IEEE. pp. 278–282.

[7] Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R., 2012. Improving neural networks by preventing co-adaptation of feature detectors. arXiv:1207.0580 [cs] URL: http://arxiv.org/abs/1207.0580. arXiv: 1207.0580.

[8] Ker, J., Wang, L., Rao, J., Lim, T., 2017. Deep learning applications in medical image analysis. Ieee Access 6, 9375–9389. Publisher: IEEE.

[9] Krizhevsky, A., 2009. Learning Multiple Layers of Features from Tiny Images. Technical Report. URL: https://www.cs.toronto.edu/~kriz/cifar.html.

[10] Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. ImageNet Classification with Deep Convolutional Neural Networks, in: Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (Eds.), Advances in Neural Information Processing Systems 25. Curran Associates, Inc., pp. 1097–1105. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[11] Mikołajczyk, A., Grochowski, M., 2018. Data augmentation for improving deep learning in image classification problem, in: 2018 International Interdisciplinary PhD Workshop (IIPhDW), pp. 117–122. doi:10.1109/IIPHDW.2018.8388338.

[12] Misra, I., Shrivastava, A., Gupta, A., Hebert, M., 2016. Cross-stitch networks for multi-task learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3994–4003.

[13] Oquab, M., Bottou, L., Laptev, I., Sivic, J., 2014. Learning and transferring mid-level image representations using convolutional neural networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1717–1724.

[14] Pan, S.J., Yang, Q., 2010. A Survey on Transfer Learning. IEEE Transactions on Knowledge and Data Engineering 22, 1345–1359. URL: http://ieeexplore.ieee.org/document/5288526/, doi:10.1109/TKDE.2009.191.

[15] Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., Liu, C., 2018. A Survey on Deep Transfer Learning, in: Kůrková, V., Manolopoulos, Y., Hammer, B., Iliadis, L., Maglogiannis, I. (Eds.), Artificial Neural Networks and Machine Learning – ICANN 2018, Springer International Publishing. pp. 270–279.

[16] Tang, B., Pan, Z., Yin, K., Khateeb, A., 2019. Recent Advances of Deep Learning in Bioinformatics and Computational Biology. Frontiers in Genetics 10. URL: https://www.frontiersin.org/articles/10.3389/fgene.2019.00214/full, doi:10.3389/fgene.2019.00214.

[17] Wang, Z., Dai, Z., Póczos, B., Carbonell, J., 2019. Characterizing and avoiding negative transfer, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 11293–11302.

[18] Yosinski, J., Clune, J., Bengio, Y., Lipson, H., 2014. How transferable are features in deep neural networks?, in: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q. (Eds.), Advances in Neural Information Processing Systems 27. Curran Associates, Inc., pp. 3320–3328. URL: http://papers.nips.cc/paper/5347-how-transferable-are-features-in-deep-neural-networks.pdf.

[19] Young, T., Hazarika, D., Poria, S., Cambria, E., 2018. Recent Trends in Deep Learning Based Natural Language Processing [Review Article]. IEEE Computational Intelligence Magazine 13, 55–75. doi:10.1109/MCI.2018.2840738.

[20] Zhang, Y., Zhang, Y., Yang, Q., 2018. Parameter Transfer Unit for Deep Neural Networks. arXiv:1804.08613 [cs, stat] URL: http://arxiv.org/abs/1804.08613. arXiv: 1804.08613.