# Tensor Decomposition for Multi-Task Deep-Learning in the context of Predictive Justice

Alexandre Audibert, Konstantin Usevich, Massih-Reza Amini, Marianne Clausel

# Tensor Decomposition for Multi-Task Deep-Learning in the context of Predictive Justice

Alexandre Audibert[†], Konstantin Usevich[‡], Massih-Reza Amini[†]
and Marianne Clausel[‡]
[†] Université Grenoble Alpes, CNRS, LIG, Grenoble
[‡] Université de Lorraine, CNRS, CRAN, Nancy

**Abstract**

In recent years, deep learning models for information retrieval have attracted a lot of attention. These models are data-hungry, necessitating large-scale training samples for learning, particularly when the goal is to associate documents with heterogeneous outputs (continuous and discrete). In this paper, we propose to apply tensor decomposition on a DL model to learn with heterogeneous outputs in the context of predictive justice. It allows us to reduce some layers by a factor of ten without sacrificing performance on the European Court of Human Rights collection.

## 1 Introduction

In this paper, we present a tensor decomposition-based compression strategy aimed at reducing the weights of a convolutional neural network that jointly learns classification and regression tasks in the context of predictive justice based on ECHR decisions. The goal is to forecast the outcome of a decision as well as the amount of money involved in a case. This issue is connected to multi-task learning [21, 17, 6], which involves learning tasks that are similar in nature. These tasks are related by the fact that, in general, an amount of money is required if and only if the case is won. For multi-task learning, many recent studies have focused on the development of large-scale neural networks based on soft and hard parameter sharing architectures [17, 19]. Moreover, when the data involves court decisions, the paucity of labeled training sets and the length of the documents constrain the use of large architectures. Semi-supervised and multi-view learning techniques have also been successfully applied to contexts with limited labeled data and multiple correlated views, as shown in multilingual document ranking [24] and Twitter sentiment classification [5], suggesting potential complementary strategies specially for predictive justice applications.

Our contributions can be summarized as follows:

- We present a tensor decomposition technique for decreasing the weights of a 1D convolutional neural network (1D-CNNs) with parameter sharing for jointly learning classification and regression tasks.

- We enforce the stability of the proposed compression algorithm, by imposing orthonormality constraints to the decomposition.
- Using a collection from the European Court of Human Rights, we demonstrate that our method allows to reduce the weight of the original model by a factor of thirty without compromising performance.

# 2 Tensor Decomposition Background

One of the major advantages of multi-task learning is its ability to improve model performance without increasing the model's capacity. Consistent with this concept of maintaining the model's cost, this section introduces Canonical Polyadic (CP) decomposition, which can be considered an extension of SVD for tensors of order greater than 2.

## 2.1 Introduction: Canonical Polyadic decomposition

CP decomposition, introduced by [13], aims at decomposing an $N$-way tensor where $N \geq 3$ into a sum of rank-one tensors. Let $\mathcal{K} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ be an $N$-way tensor. It has rank one if and only if there exists $N$ vectors $\boldsymbol{a}_{i \in [\![1,N]\!]} \in \mathbb{R}^{I_i}$ such as:

$$\mathcal{K} = \boldsymbol{a}_1 \otimes \boldsymbol{a}_2 \otimes \cdots \otimes \boldsymbol{a}_N \tag{1}$$

where $\otimes$ is the tensor (outer) product operation. A CP decomposition of a $N$-way tensor $\mathcal{K}$ is represented in Equation 2 and in Figure 1.

$$\mathcal{K} = \sum_{r=1}^{R} \boldsymbol{a}^{(1,r)} \otimes \boldsymbol{a}^{(2,r)} \otimes \cdots \otimes \boldsymbol{a}^{(N,r)}, \tag{2}$$

where $R$ is an integer. The smallest $R$ such that Equation (2) is verified, is called the rank.

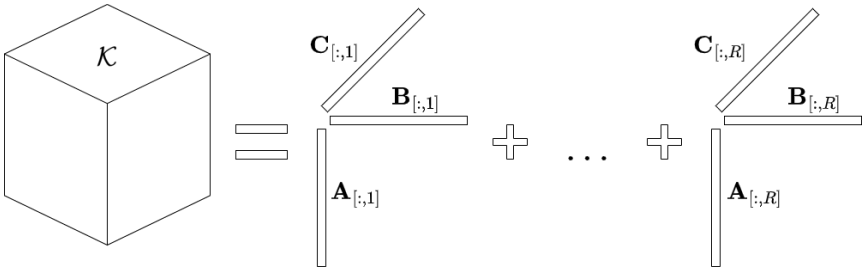


Figure 1: Canonical Polyadic decomposition

For convenience, the vectors in the decomposition (2) can be stacked into the factor matrices

$$\boldsymbol{A}^{(i)} = \begin{bmatrix} \boldsymbol{a}^{(i,1)} & \boldsymbol{a}^{(i,2)} & \cdots & \boldsymbol{a}^{(i,r)} \end{bmatrix} \in \mathbb{R}^{I_i \times R}$$

so that the decomposition (2) becomes

$$\mathcal{K}_{i_1 \ldots i_N} = \sum_{r=1}^{R} \boldsymbol{A}^{(1)}_{i_1,r} \cdots \boldsymbol{A}^{(N)}_{i_N,r} \tag{3}$$

## 2.2 Canonical Polyadic decomposition Algorithms

The goal of CP decomposition algorithms is to take a low rank $N$-way tensor as input and return a set of matrices $\{\boldsymbol{A}^{(i)}\}_{i=1}^{N}$. There are numerous different algorithms, such as CP-ALS [15] or the Tensor Power Method [2]. Recent advances, such as those discussed in [3], demonstrate that techniques like subset-based tensor approximation can further enhance the efficiency of CP decomposition algorithms, especially when dealing with large, high-dimensional tensorss. It is important to note that for each of these algorithms, the rank $R$ is provided as input. In this section, we will present the most popular one which is the ALS-CP algorithm. It is necessary to introduce some notations: $\|x\|_F$, $\odot$, $\otimes$, and $\mathcal{K}(n)$ represent, respectively, the Frobenius norm, the Khatri–Rao product, the Kronecker product, and the n-mode unfolding of $\mathcal{K}$ (unfolding according to the dimension n). For simplicity of notation, we will consider the following problem: the goal is to approximate $\mathcal{K} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ as $\sum_{i=1}^{R} \boldsymbol{a}_r \circ \boldsymbol{b}_r \circ \boldsymbol{c}_r = [\![\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!]$, where $\boldsymbol{a}_r, \boldsymbol{b}_r, \boldsymbol{c}_r$ are the columns of matrices $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$. It is possible to rewrite the problem as follows:

$$\min_{\boldsymbol{A},\boldsymbol{B},\boldsymbol{C}} \|\mathcal{K} - [\![\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!]\|_F^2 \tag{4}$$

The CP-ALS is a block-coordinate descent algorithm which updates $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ in an alternating manner. The ALS approach involves fixing $\boldsymbol{B}$ and $\boldsymbol{C}$ to solve for $\boldsymbol{A}$, then fixing $\boldsymbol{A}$ and $\boldsymbol{C}$ to solve for $\boldsymbol{B}$, and finally fixing $\boldsymbol{A}$ and $\boldsymbol{B}$ to solve for $\boldsymbol{C}$. This process repeats until a convergence criterion is satisfied. To update the matrix $\boldsymbol{A}$ it is possible to see that:

$$\mathcal{K}_{(1)} = \boldsymbol{A}(\boldsymbol{C} \odot \boldsymbol{B})^T \tag{5}$$

The problem defined in Equation 4 can be reformulated under the hypothesis of only update the matrix $\boldsymbol{A}$.

$$\min_{\boldsymbol{A}} \|\mathcal{K} - [\![\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!]\|_F^2 = \min_{\boldsymbol{A}} \|\mathcal{K}_{(1)} - \boldsymbol{A}(\boldsymbol{C} \odot \boldsymbol{B})^T\|_F^2 \tag{6}$$

The optimal solution for Equation (6) is $\boldsymbol{A} = \mathcal{K}_{(1)}((\boldsymbol{C} \odot \boldsymbol{B})^T)^\dagger$ because it provides the least squares solution to this problem. The update for the matrices $\boldsymbol{B}$ and $\boldsymbol{C}$ can be defined in a similar manner, and this Algorithm 2 is also applicable for tensors of order higher than 3.

**Algorithm 2:** CP-ALS algorithm

**Input**: Tensor $\mathcal{K}$;
Rank of the de composition R;
Initialized matrix $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$;
**while** *Until convergence* **do**
  $\boldsymbol{A} = \mathcal{K}_{(1)}((\boldsymbol{C} \odot \boldsymbol{B})^T)^\dagger$;
  $\boldsymbol{B} = \mathcal{K}_{(2)}((\boldsymbol{C} \odot \boldsymbol{A})^T)^\dagger$;
  $\boldsymbol{C} = \mathcal{K}_{(3)}((\boldsymbol{B} \odot \boldsymbol{A})^T)^\dagger$;
**return** $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$;

## 2.3 Challenge related to the CP-Decomposition

The primary challenge of CP Decomposition lies in determining the appropriate rank. In many cases, the rank is provided as input, assuming prior knowledge, which is often

not the case in practice. Furthermore, determining the rank of a tensor is known to be an NP-hard problem [12]. Additionally, CP decomposition suffers from a phenomenon known as degeneracy. This issue arises from the fact that certain tensors can be approximated arbitrarily closely by tensors of lower rank [15]. However one of the key advantage of CP composition is its uniqueness under some necessary condition.

### 2.4 Other low rank tensor decomposition

In the previous section, we focused on the CP decomposition. However, it is important to note that there is no unique definition of a low rank tensor. Other well-known approaches include the Tucker decomposition [15] and the Tensor Train [20] decomposition. These approaches introduce their own definition and representation of low rank tensors.

## 3 CP decompositions of convolutional layers

The CP decomposition approach was successfully applied to 2D-convolutional neural networks (2D-CNNs) [16, 4], where a 4-way convolutional kernel is compressed. This approach can be easily adapted to 1D-CNNs with 3-way convolutional kernels; moreover, it is more interpretable in this case. We describe it below for completeness. Let $\mathcal{K} \in \mathbb{R}^{O \times D \times N}$ be the kernel of a conv1D with $O$ and $D$ being the number of the output and input channel respectively, and $N$ the window dimension for $N$-grams. Then $\mathcal{K}$ can be approximated by with CP decomposition as follows:

$$\mathcal{K}_{o,d,n} = \sum_{r=1}^{R} \boldsymbol{A}^{(1)}_{o,r} \boldsymbol{A}^{(2)}_{d,r} \boldsymbol{A}^{(3)}_{n,r} \tag{7}$$

where $\boldsymbol{A}^1 \in \mathbb{R}^{O \times R}$, $\boldsymbol{A}^2 \in \mathbb{R}^{D \times R}$, $\boldsymbol{A}^3 \in \mathbb{R}^{N \times R}$. Our kernel is approached by a tensor of rank at most $R$.

In our study, we denote by $\boldsymbol{X} \in \mathbb{R}^{T \times D}$ a document of length $T$ with the embedding dimension $D$. Then a one-dimensional convolution (conv1D) can be written as follows:

$$\boldsymbol{E}_{t,o} = \sum_{d=1}^{D} \sum_{n=1}^{N} \boldsymbol{X}_{t+n-\delta,d} \mathcal{K}_{o,d,n}, \tag{8}$$

where $\delta$ is the "half-width" window $\frac{N-1}{2}$ and $\boldsymbol{E} \in \mathbb{R}^{T \times O}$ is the output of the conv1D. Note that we talked previously about approximation, and now it is a decomposition. Thanks to the CP decomposition (7), Equation (8) can be reformulated as follows :

$$\boldsymbol{E}_{t,o} = \sum_{r=1}^{R} \boldsymbol{A}^{(1)}_{o,r} (\sum_{d=1}^{D} \boldsymbol{A}^{(2)}_{d,r} (\sum_{n=1}^{N} \boldsymbol{X}_{t+n-\delta,d} \boldsymbol{A}^{(3)}_{n,r})) \tag{9}$$

Hence the computation of $\boldsymbol{E}_{t,o}$ naturally decomposes into consecutive operations:

$$\mathcal{T}^{(1)}_{t,i,r} = \textstyle\sum_{n=1}^{N} \boldsymbol{X}_{t+n-\delta,i} \boldsymbol{A}^{(3)}_{n,r}, \tag{10}$$

$$\mathbf{T}^{(2)}_{t,r} = \textstyle\sum_{d=1}^{D} \mathcal{T}^{(1)}_{t,d,r} \boldsymbol{A}^{(2)}_{d,r}, \tag{11}$$

$$\mathbf{E}_{l,o} = \textstyle\sum_{r=1}^{R} \mathbf{T}^{(2)}_{t,r} \mathbf{A}^{(1)}_{o,r}. \tag{12}$$

The Equations (10), (11) and (12) can be computed like in the case of conv2D with respectively a window size $(D, 1)$ and R output channels.

### 3.1 Interpretability on 1D-CNNs for text

A benefit of CP decomposition applied to conv1D is the ability that it gives to interpret the results, which is more difficult to understand in the case of conv2D for image classification. Indeed the first convolutional step computes $R$ weighted sums of $N$-grams (Equation 10). The results of this layer is $R$ $D$-dimensional embeddings of each $N$-grams. After that another convolutional operation is applied, the output can be interpreted as an importance scores per $D$-dimensional embeddings which results to $R$-dimensional representation of the original $D$-grams (Equation 11). Finally, the last step makes $O$ different weighted sum of the previously computed values (Equation 12).

## 4 Our Approach

We adapted the simple yet efficient 1D-CNNs model proposed in [14] to the scenario of multi-task learning, as the ECHR data collection is not large and contains long documents making it difficult to employ other complicated NN algorithms. This model contains three main parts and it is shown in Figure 2. The first part consists of two embedding layers, the first of which initializes embeddings with pre-trained word2vec (static channel) and the second one initializes embeddings with pre-trained word2vec and then fine-tunes them during training (non-static channel). The activation ReLU is used in the second part, which employs three parallel convolutions for 3,4, and 5 $N$-grams. This part is referred to as convolutional-block in the following. A max-pooling over time is then used to keep the higher value of each features maps. The last part is made up of two fully connected layers with dropout and sigmoid outputs for the classification task and the linear activation function for regression. This architecture is referred to as hard-sharing in the literature [17, 6] .

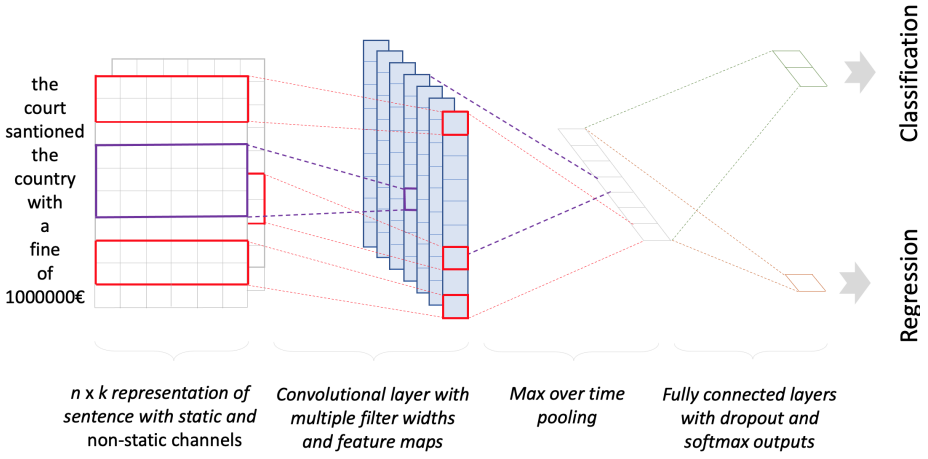


Figure 2: A 1D-CNNs architecture for multi-task learning

### 4.1 Fused CP decomposition

We note convolutional-CP($\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$) the convolution operation where the matrix $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ are the kernel of convolution described by Equations (10)–(12). Our approach is to ap-

ply the CP decomposition on each conv1D in the convolutional-block (second part). This process works as follows. Instead of applying CP Decomposition on each 1D kernel in a separate manner, we concatenate the three 1D kernels along the third-mode to obtain a new tensor of shape $O \times D \times (N_1 + N_2 + N_3)$. The CP decomposition applies on this bigger tensor giving three matrices of sizes $\boldsymbol{A}^{(1)} \in \mathbb{R}^{O \times R}$, $\boldsymbol{A}^{(2)} \in \mathbb{R}^{D \times R}$, $\boldsymbol{A}^{(3)} \in \mathbb{R}^{(N_1+N_2+N_3) \times R}$. After that the matrix $A^{(3)}$ can be cut into three matrices $\boldsymbol{A}_1^{(3)} \in \mathbb{R}^{D_1 \times R}$, $\boldsymbol{A}_2^{(3)} \in \mathbb{R}^{D_2 \times R}$ and $\boldsymbol{A}_3^{(3)} \in \mathbb{R}^{D_3 \times R}$.
The three separate 1D kernels can be computed as before : convolutional-CP$_i(\boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \boldsymbol{A}_i^{(3)})$. The important thing is that the three parallel conv1D have the same convolutional operation represented by the matrix $\boldsymbol{A}^{(1)}$ and $\boldsymbol{A}^{(2)}$. It is possible to notice that our approach is different than the standard approach used in ML because we concatenate tensor with different sizes along a mode instead of stacking tensors with the same size [9, 26].

## 4.2 Algorithm for CP Decomposition with Orthogonality constraint

Indeed, in most of the cases, an optimal solution to approximate a $N$-way tensor with $N \geq 3$ by a tensor with a lower rank does not exist i.e the cost function in the approximation problem may only have an infimum and not a minimum. This issue seems to have negative impacts on the fine-tuning step as reported in [16]. This problem was also highlighted in [4]. The proposed approach overcomes this issue by using an iterative fine-tuning.
To this end, we propose to use an Alternating Least Squares algorithm which imposes to have a semi-orthogonal matrix $\boldsymbol{A}^{(j)}$. This constraint enforces CP decomposition to have an optimal solution [22]. Our algorithm bears similarity to the standard CP-ALS, which updates one matrix per iteration based on least square approach. In our CP-ALS with orthogonality, if our matrix $\boldsymbol{A}^{(i)}$ does not have to be orthonormal the update is identical to the standard CP-ALS. In the other case we use the toolbox `https://pymanopt.org`Pymanopt [23] for optimization on Riemannian manifolds (in our case the Stiefel manifold) to find a semi-orthogonal matrix $\boldsymbol{A}^{(j)}$ minimizing $\left\|\boldsymbol{A}^{(j)} - \boldsymbol{X}_{(j)}((\boldsymbol{A}^{(3)} \odot \boldsymbol{A}^{(1)})^T)^\dagger\right\|_F$ in the case where j = 2. Orthonormal constraint is applied on matrix used in Equations (11) or (12) intuitively creates more different features and fights redundancy.

**Algorithm 5:** CP-ALS with orthogonality constraint on the second dimension

**Data:** Tensor $\mathcal{X} \in \mathbf{R}^{I_1 \times I_2 \times I_3}$, integer $R$, integer $i \in [1, 3]$ and $I_i \geq R$
**Result:** Generate $\mathbf{A}^{(1)} \in \mathbf{R}^{I_1 \times R}$, $\mathbf{A}^{(2)} \in \mathbf{R}^{I_2 \times R}$, $\mathbf{A}^{(3)} \in \mathbf{R}^{I_3 \times R}$
with $\mathbf{A}^{(i)}$ column-wise orthonormal
Initialize $A^{(j)} \in \mathbf{R}^{I_i, R}$ for j $\in$ [1, 3];
Here i is supposed to be equal to 2 but it can be generalized.
**while** *Criterion* **do**
$\mathbf{A}^{(1)} = \mathbf{X}_{(1)}((\mathbf{A}^{(3)} \odot \mathbf{A}^{(2)})^{(T)})^\dagger$;
$\mathbf{A}^{(2)} = \text{argmin}_{\mathbf{A} \in V_{I_2}(\mathbf{R}^R)} \left\{\left\|\mathbf{A} - \mathbf{X}_{(2)}((\mathbf{A}^{(3)} \odot \mathbf{A}^{(1)})^T)^\dagger\right\|_F\right\}$;
$\mathbf{A}^{(3)} = \mathbf{X}_{(3)}((\mathbf{A}^{(2)} \odot \mathbf{A}^{(1)})^T)^\dagger$;
**return** $\mathbf{A}^{(j)} \in \mathbf{R}^{I_j, R}$ for j $\in$ [1, 3];

### 4.3 Computational and storage complexity

The complexity of standard conv1D is $\mathcal{O}(TODN)$ while the complexity of the conv1D with CP decomposition is the sum of the complexity of the three Equations (10)–(12) with the respective complexities $\mathcal{O}(TNRD)$, $\mathcal{O}(TRD)$, $\mathcal{O}(TRO)$. In the majority of the case D $\gg$ N, that is why it is possible to reorder 10 and 11 to obtain a complexity of $\mathcal{O}(TDRN+TRN+TRO)$. Based on these remarks, the comparison of computational and storage complexities for the two approaches is provided in Table 1.

Table 1: Computational and storage complexity on convolutional-block

| **Approach** | **Computational complexity** | **Storage complexity** |
|---|---|---|
| Parallel's conv1D | $\mathcal{O}(12TOD)$ | $12DO + 3O$ |
| Parallel's conv-CP | $\mathcal{O}(TR(12D + 12 + 3O))$ | $R(12 + 3D + 3O) + 3O$ |
| Conv-CP fusion | $\mathcal{O}(TR(12D + 12 + 3O))$ | $R(12 + D + O) + 3O$ |

## 5 Experiments and results

### 5.1 Dataset

In our experiments, we use the collection described in [7][1]. It was built on the basis of documents provided by the European Court of Human Rights. This dataset was designed primarily for classification as a binary task with the aim of predicting whether or not there has been a breach of human rights legislation. These documents also include a monetary sum (referred to as quantum) related to cases of legal violations, which is relevant to predictive justice. As a result, we used regular expressions to extract quantum and then applied a log function to normalize these values between 0 and 1. In order to have the same currency, we removed documents prior to 2002 (before the apparition of Euro).

### 5.2 Experiments details

Our preprocessing is very similar to [11] and it consists in removing stop words and infrequent words with a document frequency less than 5; and applying lemmatisation. For the non-static channel of the embedding layer part. We initialized word embeddings with law2vec[2] which is a word2vec model trained on law documents. The output-channel is equal to 64 in our experiments against 100 in the original paper [14]. The embedding and the output-channels were chosen to avoid useless over-parametrization of the model. The model is trained by joint-learning as in [8] by minimizing a weighted sum of MAE and binary cross-entropy which writes:

$$\mathcal{L} = \lambda \mathcal{L}_{BCE} + (1 - \lambda)\mathcal{L}_{MAE} \tag{13}$$

where $\lambda$ is set to 0.85 in our experiments. For the hyperparameter, we used the optimizer Adam, a batch size of 16, learning rate equal to $1e^{-4}$ for model without decom-

[1] https://archive.org/details/ECHR-ACL2019
[2] https://archive.org/details/Law2Vec

Table 2: Single-Task Learning

| Model | F1 score ($\pm$ std) | MAE ($\pm$ std) | Reduction weight |
|---|---|---|---|
| 1D-CNN | $81.44 \pm 0.14$ | $0.3680 \pm 0.0016$ | 1.00 |
| 1D-CNN with CP-ALS | $80.84 \pm 0.37$ | $0.3690 \pm 0.0019$ | 11.45 |
| 1D-CNN with CP-ALS-ortho 3 | $80.81 \pm 0.31$ | $0.3697 \pm 0.0014$ | 11.45 |
| 1D-CNN with CP-ALS-ortho 2 | $80.81 \pm 0.32$ | $0.3702 \pm 0.0015$ | 11.45 |
| 1D-CNN with CP-ALS-Fusion | $80.03 \pm 0.45$ | $0.3743 \pm 0.0030$ | 30.74 |
| 1D-CNN with CP-ALS-ortho 3 | $80.38 \pm 0.40$ | $0.3729 \pm 0.0029$ | 30.74 |
| 1D-CNN with CP-ALS-ortho 2 | $80.69 \pm 0.18$ | $0.3717 \pm 0.0024$ | 30.74 |

position and $1e^{-5}$ for model with decomposition. Finally, the rank $R$ in CP decomposition was set by cross-validation and it was in most of the time equal to 16. For the single task model and multi-task model we compare seven approaches. Moreover we also compare the CP-ALS with orthogonality constraint and the standard CP-ALS algorithm [15].

1. Standard 1D-CNNs as proposed in [14].
2. 1D-CNNs with CP-ALS model where CP-ALS algorithm is applied individually to each conv1D.
3. 1D-CNNs with CP-ALS-ortho (2 or 3) model where CP-ALS-ortho is applied individually to each conv1D. The orthogonality constraint is applied on the second conv1D or on the third conv1D in the decomposition.
4. 1D-CNNs with CP-ALS-Fusion model where CP-ALS algorithm is used with fusion approach.
5. 1D-CNNs with CP-ALS-ortho (2 or 3) fusion model where CP-ALS-ortho is used with fusion approach.

Table 3: Multi-Task Learning

| Model | F1 score ($\pm$ std) | MAE ($\pm$ std) | Reduction weight |
|---|---|---|---|
| 1D-CNN multi-task | $81.60 \pm 0.17$ | $0.3644 \pm 0.0033$ | 1.00 |
| 1D-CNN with CP-ALS | $80.72 \pm 0.25$ | $0.3660 \pm 0.0042$ | 11.45 |
| 1D-CNN with CP-ALS-ortho 3 | $81.02 \pm 0.22$ | $0.3638 \pm 0.0019$ | 11.45 |
| 1D-CNN with CP-ALS-ortho 2 | $80.87 \pm 0.18$ | $0.3637 \pm 0.0028$ | 11.45 |
| 1D-CNN Fusion ALS | $79.51 \pm 0.21$ | $0.3641 \pm 0.0019$ | 30.74 |
| 1D-CNN Fusion ortho conv3 | $79.88 \pm 0.40$ | $0.3660 \pm 0.0022$ | 30.74 |
| 1D-CNN Fusion ortho conv2 | $80.32 \pm 0.35$ | $0.3641 \pm 0.0036$ | 30.74 |

The metrics used to compare these approaches are Macro-F1 like in [7] for the classification and MAE for the regression task.

## 5.3 Results

The reduction weight in Tables 3 and 2 represents reduction weight for the convolutional layers in the convolutional-block. We note that results on classification are

comparable to the state-of-the-art reported in [7] on Table 4. The results obtained in Table 2 show that the orthogonality constraint has no impact on the performance if the convolutional layers in convolutional-block are treated separately. However when this approach is used with the fusion method, the orthogonality constraint on the second layer gives similar results on classification and regression task than the standard approach with approximately three times less parameter on the convolutional-block. Compared to the results of [14] our best approach: 1D-CNNs with CP-ALS-ortho 2 obtains a loss of approximately 1 in Macro-F1 and 0.005 on the MAE but with a reduction factor of more than 30 of the convolutional-block.

Table 4: Single-Task Learning [7]

| **Approach** | **Macro-F1 ($\pm$ std)** |
|---|---|
| HIER-BERT [10] | $80.1 \pm 1.1$ |
| HAN [27] | $80.2 \pm 2.7$ |
| BIGRU-ATT [18] | $78.9 \pm 1.9$ |
| BOW-SVM [1] | $70.9 \pm 0.0$ |

The results obtained in Table 3 show that orthogonality constraints seem to have a positive effect on the fine-tuning. Without fusion approach, the orthogonality constraints on the third layer obtains results with a loss of 0.6 Macro-F1 compare to standard 1D-CNNs multi-task and no loss of MAE, with ten times less parameters on the convolutional-block. With the fusion approach, as in single task the orthogonality constraint applied on the second layer gives the best result. The comparison of results before and after compression shows that with the CP decomposition, the classification and regression performance are slightly affected with thirty times fewer parameters than the original model; and that the classification performance is competitive when compared to the state of the art.

# 6 Conclusion

This paper addresses the challenge of reducing the weights of 1D convolutional neural networks using CP Decomposition for multi-task learning in predictive justice. Our findings demonstrate that it is possible to reduce the weights by a factor of 30 without compromising performance. Inspired by these results, our future work will explore tensor decomposition with multi-task learning using more complex models on larger datasets and broader multi-task architectures, where a shared model is used for all tasks alongside task-specific modules, as studied in [25].

# References


[1] Nikolaos Aletras, Dimitrios Tsarapatsanis, Daniel Preoţiuc-Pietro, and Vasileios Lampos. Predicting judicial decisions of the european court of human rights: A natural language processing perspective. *PeerJ Computer Science*, 2:e93, 2016.

[2] Genevera Allen. Sparse higher-order principal components analysis. In Neil D. Lawrence and Mark Girolami, editors, *Proceedings of the Fifteenth International*

*Conference on Artificial Intelligence and Statistics*, volume 22 of *Proceedings of Machine Learning Research*, pages 27–36, La Palma, Canary Islands, 2012. PMLR.

[3] Hesam Amoualian, Wei Lu, Eric Gaussier, Georgios Balikas, Massih-Reza Amini, and Marianne Clausel. Topical coherence in lda-based models through induced segmentation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1799–1809, 2017.

[4] Marcella Astrid and Seung-Ik Lee. Cp-decomposition with tensor power method for convolutional neural networks compression. In *2017 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pages 115–118. IEEE, 2017.

[5] Georgios Balikas and Massih-Reza Amini. Twise at semeval-2016 task 4: Twitter sentiment classification. In *10th International Workshop on Semantic Evaluation*, pages 85–91, 2016.

[6] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[7] Ilias Chalkidis, Ion Androutsopoulos, and Nikolaos Aletras. Neural legal judgment prediction in english. *Association for Computational Linguistics*, 2019.

[8] Jing Chen, Long Cheng, Xi Yang, Jun Liang, Bing Quan, and Shoushan Li. Joint learning with both classification and regression models for age prediction. In *Journal of Physics: Conference Series*, volume 1168, page 032016. IOP Publishing, 2019.

[9] Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. Multi-head attention: Collaborate instead of concatenate. In *International Conference on Representation Learning - ICLR*, 2021.

[10] Devlin. Bert: Pre-training of deep bidirectional transformers for language understanding. *In North American Association for Com- putational Linguistics (NAACL)*, 2018.

[11] Jiachen Du, Lin Gui, Ruifeng Xu, and Yulan He. A convolutional attention model for text classification. In *National CCF conference on natural language processing and Chinese computing*, pages 183–195. Springer, 2017.

[12] Johan Håstad. Tensor rank is np-complete. In *Automata, Languages and Programming: 16th International Colloquium Stresa, Italy, July 11–15, 1989 Proceedings 16*, pages 451–460. Springer, 1989.

[13] F.L. Hitchcock. The expression of a tensor or a polyadic as a sum of products. *J. Math. Phys.*, 6:164–189, 1927.

[14] Yoon Kim. Convolutional neural networks for sentence classification. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October 2014. Association for Computational Linguistics.

[15] Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. *SIAM Review*, 51:455–500, 2009.

[16] Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, and Victor Lempitsky. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. *Computer Science*, 2014.

[17] Liu. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496, Florence, Italy, July 2019. Association for Computational Linguistics.

[18] Bingfeng Luo, Yansong Feng, Jianbo Xu, Xiang Zhang, and Dongyan Zhao. Learning to predict charges for criminal cases with legal basis. 2017.

[19] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3994–4003, 2016.

[20] I. V. Oseledets. Tensor-train decomposition. volume 33, pages 2295–2317, 2011.

[21] Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. Latent multi-task architecture learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4822–4829, 2019.

[22] Mikael Sørensen, Lieven De Lathauwer, Pierre Comon, Sylvie Icart, and Luc Deneire. Canonical polyadic decomposition with a columnwise orthonormal factor matrix. *SIAM Journal on Matrix Analysis and Applications*, 33(4):1190–1213, 2012.

[23] James Townsend, Niklas Koep, and Sebastian Weichwald. Pymanopt: A python toolbox for optimization on manifolds using automatic differentiation. *Journal of Machine Learning Research*, 2016.

[24] Nicolas Usunier, Massih-Reza Amini, and Cyril Goutte. Multiview semi-supervised learning for ranking multilingual documents. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*, page 443–458, 2011.

[25] Sen Wu, Hongyang R Zhang, and Christopher Ré. Understanding and improving information transfer in multi-task learning. In *International Conference on Representation Learning - ICLR*, 2020.

[26] Yongxin Yang and Timothy Hospedales. Deep multi-task representation learning: A tensor factorisation approach. In *International Conference on Representation Learning - ICLR*, 2017.

[27] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, San Diego, California, June 2016. Association for Computational Linguistics.

**Temporary page!**

LaTeX was unable to guess the total number of pages correctly. As there was some unprocessed data that should have been added to the final page this extra page has been added to receive it.

If you rerun the document (without altering it) this surplus page will go away, because LaTeX now knows how many pages to expect for this document.