Compact and Efficient Multitask Learning in Vision, Language and Speech

Mohammed Al-Rawi and Ernest Valveny
Computer Vision Center (CVC), UAB, Barcelona, Spain
ms.alrawi@gmail, ernest@cvc.uab.es

Abstract

Across-domain multitask learning is a challenging area of computer vision and machine learning due to the intra-similarities among class distributions. Addressing this problem to cope with the human cognition system by considering inter and intra-class categorization and recognition complicates the problem even further. We propose in this work an effective holistic and hierarchical learning by using a text embedding layer on top of a deep learning model. We also propose a novel sensory discriminator approach to resolve the collisions between different tasks and domains. We then train the model concurrently on textual sentiment analysis, speech recognition, image classification, action recognition from video, and handwriting word spotting of two different scripts (Arabic and English). The model we propose successfully learned different tasks across multiple domains.

1. Introduction

Learning the representation of different tasks across multiple domains via one model is a challenging machine learning problem. Such compact multitasking models can find huge applications in restricted computing scenarios, for example, mobile and autonomous driving applications; by helping reduce power consumption, memory and hardware usage. Regardless of the application scenario of this one (universal) model, the machine learning community has always been interested in comparing the performance of machine intelligence with the human brain. While the human brain is highly capable of learning various tasks across multiple domains, computer vision and machine learning approaches are still lagging when it comes to using one model to learn several tasks. Until recently, works dedicated to investigate one model learning usually incorporate Multi-Model building blocks (modular nets) from multiple domains. One of the most prominent works in this direction appeared in "One Model To Learn Them All" [16]. Furthermore, Multitask Learning (MTL) usually involves optimizing more than one loss function [32], *e.g.* one loss for each model, which increases the search space complexity and parametric tuning. In this work, we opted to try a challenging multitask approach by using only one deep learning model, one loss function, and one output layer for all tasks across multiple domains. Furthermore, to enable the OneModel accommodate to different tasks across multiple domains, we use a text embedding in which each task or category is represented by a set of unique words. Using such representation, the model can learn if a word that comes from the audio sensory is in a different encoding space than the same word that comes from the text or vision sensories. Therefore, one model can learn if a word mimicking a task is coming from the speech sensory in a different way when the same word comes from text or visual sensories. Our approach depends on the fact that there are different ways in which humans and / or machines can perceive the world. For example, the word "car" can be sensed from a car image, a speech form of a person saying "car", text containing the "car" word, or a handwritten of the word "car". We are able in this work to achieve such cognition, perception and discrimination by adding a newly generated unique hashcode to each word; for which the hashcodes are divided into several groups that mimic vision, language and speech, or even down to the task level. This hash function copes with the biological brain; for example, when the sensory retina sends the vision signal to the lateral geniculate nucleus and then to the visual cortex via the optic nerve, and the auditory sensory region transmits the speech / sound signal to the primary auditory cortex via the vestibulocochlear nerve. Nonetheless, although the human brain is highly modular and has different regions for vision, text and sound, we are considering a major challenge in this work using only one model (OneModel) that handles the various information from the sensory layer. The OneModel that we propose, illustrated in Figure 1, may enable machines to have a conscious at the higher level of the learning model, *e.g.* the output layer.

2. Related Works

Although MTL improves generalization by taking advantage of domain-specific information contained in the train-

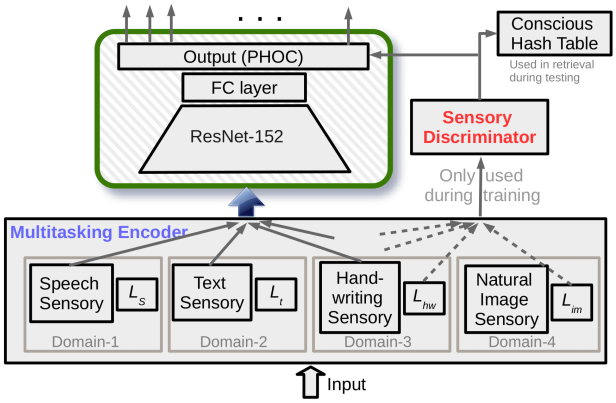

Figure 1. The proposed OneModel architecture. L_s, L_t, L_{hw}, and L_{im} are the ground-truth data of speech, text, and vision (handwriting and images) such that each of their entries can be represented as a string containing alphanumeric and non-alphanumeric symbols. Alphabets can be of different languages and / or scripts.

ing signals of related tasks [5, 6], it still needs further investigation if it is to be considered across multiple domains. In the context of Deep Learning, MTL is usually implemented via layer sharing, *e.g.* using either hard or soft parameter-sharing at the hidden layers [32]. The hard parameter-sharing is applied by sharing the hidden layers between all tasks and then using several task-specific output layers, *i.e.* one layer per task. It has been shown in [4] that using hard parameter sharing leads to better generalization, by reducing the risk of over-fitting. Soft parameter sharing, on the other hand, is achieved by sharing intra connected Multi-Model blocks, *i.e.* each model has its own parameters of the task, such that the distance between the parameters of the model is then regularized [32]. However, similar to hard parameter sharing, soft parameter sharing still needs task-specific output layers.

The MTL literature also shows slight deviations from soft and/or hard layer sharing, which has only been applied to one domain learning problems. As an example, using matrix priors as inputs to the task-specific output layers to allow the model to learn the relationship between tasks [18]. Other MTL approaches are based on dual task learning form a single dataset. For example, model level dual learning, which leverages task duality to redesign the model architectures for both the primal and dual task [42] helped achieving better performance in machine translation task. However, the proposed dual-learning model has only been applied to one dataset at a time and is consistent with previous MTL works by implementing two loss functions, one for the primal task and another for the dual task. In addition, the dual-learning model is built using two building block models. For example, it has been used for textual sentiment analysis for which the primal task is sentiment classification that aims to justify whether a natural language sentence has positive or negative sentiment. The dual task is language modeling of a sentence conditioned on a given positive or negative sentiment label [42]. Accordingly, MTL restrictions, particularly in deep learning, appear to be primarily applied to one domain, based on multiple model blocks and the use of task-specific output layers. Nonetheless, although the work presented in [16] is inspirational for building 'one model' based on multiple building blocks, each dedicated to a single task, other works have implemented ideas that are slightly similar in nature / purpose to ours, by placing a standalone graph layer on top of any feed-forward architecture [9]. In the latter work, the authors developed a model that allows encoding of flexible relations between labels.

3. Methods

3.1. The Multitasking Encoder

We use a word embedding technique to represent the output of each task regardless of its domain. In this work, we adopt the Pyramidal Histogram of Characters (PHOC; [3]) to convert a string, that mimics the task and its sub-labels, to a vector that can be used to supervise the learning. The PHOC algorithm, $\psi_m(l)$, simply divides an input word l into a few regions (m levels) and then finds the histogram of characters falling within each region [3]. The concatenation of all these regions/ histograms gives the PHOC vector ζ, which is the label that can be used to supervise the learning. This can be formulated as follows:

$$\zeta = \psi_m(l), \tag{1}$$

where m is the PHOC level, l is the word label. ζ has the cardinality of $(m^2+m-2)/(2N_s)$, and N_s is the number of symbols used for the word-label set. For English alphabets $N_s = 26$. In this work, we propose a novel PHOC representational approach to denote across-domain, inter and intra class categories, which can be formulated as follows:

$$\eta = \psi_m(\cup_{k=1}^{n} L_k), \tag{2}$$

where l_k is a set denoting the alphabet used to denote each label of the k_{th} task (could be within or across domains), its entries can also be alphanumerics, and $k = 1, 2, ..., n$ with n denoting the number of tasks. For example, to model a system that learns English and Arabic languages, and tasks related to these scripts, η has a total of 54 symbols (Arabic has 28 alphabets). In this work, we use up to five PHOC levels, *i.e.* PHOC levels used are {2 ,3, 4, 5}. It is possible that different tasks share the same alphanumeric set, unless the models must learn tasks for different languages.

3.2. The Sensory Discriminator

Sensory discrimination is one of the vital components of this work. It aims to distinguish between different tasks by

adding uniqueness to each task and / or domain. Each class and / or task has a name representing it, *e.g*. each speech signal denotes some spoken word and each handwritten image refers to some word. Hence, every problem R_k contains two groups 1) the data source S_k, which may take the form of images, signals, or text, *etc*.; and 2) the corresponding ground-truth L_k, which contains labels in the form of text. These groups can be formulated as:

$$\begin{aligned} S_k &= [s_1, s_2, ..., s_q], \\ L_k &= [l_1, l_2, ..., l_q], \end{aligned} \tag{3}$$

where q is the total number of samples in the problem R_k, to simplify notations we dropped the subscript k from q. If there is no redundancy in the data, S_k is normally a finite set. The situation for L_k is different since it can not be represented as a finite set, because each item in L_k usually contains some text that mimics the task denoted in words, tags, labels, *etc*. Hence, let the F_k be a finite set of (unique) items from L_k, which we denote as:

$$F_k = \{f_1, f_2, ..., f_p\}, \tag{4}$$

where p is the number of labels, with $p \leq q$. Clearly, it is possible that two different tasks and / or domains to use the same labels, leading to confusion during learning and will therefore result in poor performance. Humans can easily distinguish entities according to the type of sensory. To resolve this problem in machine / deep learning, each label / keyword in F_k is appended, or modulated, by a unique hashcode. This can be formulated as follows:

$$G_k = \{g_1, g_2, ..., g_{p_k}\}, \tag{5}$$

where

$$g_{p_k} = \gamma(f_{p_k}, \nu_k), \tag{6}$$

such that γ is the sensory discriminator transform, and ν_k is the task identifier of the problem l_k. Hence, the output layer will have the following representation:

$$\zeta = \psi_m(\cup_{k=1}^{n} G_k). \tag{7}$$

To give a simple example of using the γ discriminator transform, one can simply use the concatenation of f_{p_k} and ν_k as follows:

$$\gamma_{||}(f_{p_k}, \nu_k) = f_{p_k}||\nu_k, \tag{8}$$

and let us assume that the word 'sky' is used when having a speech signal of someone saying 'sky' ($\nu_k = \texttt{speech}$), then:

$$\begin{aligned} \gamma_{||}(\texttt{sky}, \texttt{speech}) &= \texttt{sky}||\texttt{speech} \\ &= \texttt{skyspeech}. \end{aligned} \tag{9}$$

The drawback of this simple concatenation is that it does not provide sparsity in the words / labels / tags when these tags are used to supervise the learning. This is because the partial word denoting the task and / or domain will be repeated across several labels in the domain, and this will degrade the learning and eventually affect the model's performance. To overcome this problem, we propose to use `DJB2` hashing algorithm [31] that has improved hashing uniqueness without the need to use a collision-resistant function. Hash tables have $O(1)$ average search times, making them efficient data structure to use for caching, indexing, and other time-critical operations. The `DJB2` algorithm, which has a performance equivalent to the generic alphanumeric MD5 hashing but results in lower numeric hashcode lengths, is currently used in quite a few information retrieval systems. Hence, we can write the discriminator transform as follows:

$$\gamma_{\texttt{DJB2}}(f_{p_k}, \nu_k) = f_{p_k}||\texttt{DJB2}(f_{p_k}||\nu_k), \tag{10}$$

and using `DJB2` for the above example gives:

$$\begin{aligned} \gamma_{\texttt{DJB2}}(\texttt{sky}, \texttt{speech}) &= \texttt{sky}||\texttt{DJB2}(\texttt{sky}||\texttt{speech}) \\ &= \texttt{sky}8246919618043377881. \end{aligned} \tag{11}$$

Clearly, `DJB2(sky||speech)` results in the hashcode $8,246,919,618,043,377,881$; which has 19 digits. If the length of this hashcode poses a problem, it can be further reduced using the modulo operator with the first prime number greater than the size of words / labels of the dictionary of the task(s) and / or domain(s). To explain this in a simple but trivial example, suppose that the labels' dictionary used contains the 127 English stop words, then, the first prime number after 127 is 131. In this case, $8,246,919,618,043,377,881 \bmod 131 = 65$. Furthermore, the hashcodes will be grouped into a task-based dictionary as follows:

$$\mathcal{D} = \{d_1, d_2, ..., d_n\}. \tag{12}$$

This means that each task / domain will have its own set of dictionary of hashcodes, which will provide robust distinction between both tasks and / or domains. In addition, this hashing will enable us to categorize and / or retrieve each query according to the task to which it belongs. We ought to mention that we did not use the hashing algorithm in for the handwriting domain.

3.3. Datasets

Table 2 shows the number samples distributed across multiple domains. As is evident, the datasets used in this work are unbalanced across tasks and domains, another challenge that has been raised in this work.

Original movie review	Review after cleaning
This has always been one of my favourite movies, and will always be. Over the last few years I have become a 50\'s / 60\'s Sci-fi freak, trying to collect all of the better ones that were made back then. I love lots of things about them from how corny they could be to how technically correct some of them were. The great colours and the sets get me going too. It\'s a pity when they re-make some of these good old movies; they nearly always stuff it up, - just look at the recent re-do of The day the Earth stood still, it\'s utter garbage!! Forbidden Planet is one of the benchmark space films of all time, and now they\'re trying to re-make it too, and I shudder to think what the new one will be like! To my mind, some things, such as fantastic classic movies, should just be left alone to be what they are, classic examples of great attempts at telling simple stories, and giving people a thrill in the process. Once they add all the techno-crap that we have available now, the film just seems to be more dog-meat from the Hollywood sausage factory, - nothing special at all. By the way, I notice that the astronauts\' uniforms in Forbidden Planet were also used for "Queen of Outer Space"! That just tells you that the budgets were a bit lower back then, doesn\'t it? Hey, less money and better films, hmmm.... Great performances in this movie from Leslie Nielsen, in a serious role, and Anne Francis, Walter Pidgeon (who has always been one of my favourite actors), Earl Holiman, and of course Robby the Robot! The special effects are fantastic, and the storyline is not too far-fetched. This is a great sci-fi experience!	this has always been one of my favourite movies, and will always be. over the last few years i have become a 50 60 science fiction freak, trying to collect all of the better ones that were made back then. i love lots of things about them from how corny they could be to how technically correct some of them were. the great colours and the sets get me going too. it is a pity when they re make some of these good old movies; they nearly always stuff it up, just look at the recent re do of the day the earth stood still, it is utter garbage exclamation_mark exclamation_mark forbidden planet is one of the benchmark space films of all time, and now they are trying to re make it too, and i shudder to think what the new one will be like exclamation_mark to my mind, some things, such as fantastic classic movies, should just be left alone to be what they are, classic examples of great attempts at telling simple stories, and giving people a thrill in the process. once they add all the techno crap that we have available now, the film just seems to be more dog meat from the hollywood sausage factory, nothing special at all. by the way, i notice that the astronauts uniforms in forbidden planet were also used for "queen of outer space" exclamation_mark that just tells you that the budgets were a bit lower back then, does not it questionmark hey, less money and better films, hmmm.... great performances in this movie from leslie nielsen, in a serious role, and anne francis, walter pidgeon (who has always been one of my favourite actors), earl holiman, and of course robby the robot exclamation_mark the special effects are fantastic, and the storyline is not too far fetched. this is a great science fiction experience exclamation_mark.

Table 1. One movie review before and after cleaning.

3.3.1 IMDB dataset

This is a dataset for binary sentiment classification containing a total of $50,000$ image movie (English) reviews [19], where the labels are balanced with $25,000$ positive reviews and $25,000$ negative reviews. Furthermore, it basically has a set of $25,000$ highly polar movie reviews for training, and $25,000$ for testing. One of the key aspects of the IMDB dataset is that each review has several sentences. The train and test sets contain a disjoint set of movies and no more than 30 comments are included for any particular movie. To eliminate neutral reviews, a negative review has a score ≤ 0.4 and a positive review has a score ≥ 0.7.

3.3.2 Speech Dataset

We use TensorFlow Speech Commands Datasets Version 1.0 [41, 40] (SPCHD). The dataset has one-second long utterances of 30 short (English) words collected from thousands of different people. It includes thirty different speech commands and we only use $64,721$ samples for training and $2,489$ for testing.

3.3.3 Handwriting datasets

We use the Institute of Informatics and Applied Mathematics English handwriting dataset, which is known as the IAM-dataset [22, 21], and IFN/ENIT (Arabic) dataset [27, 28]. Both IAM and IFN datasets are multi writers datasets. For the IAM-dataset, we use both the training and validation sets for training, similar to [34]. To cope with the IAM previous works [17], we remove the stop words from the query. For the IFN dataset, we partition the data in such a way that 75% of the samples used for training and the remaining 25% for testing. This data partitioning has been adopted by several previous word-spotting and recognition works, see for example [35, 36, 34, 37, 2]. We note that the IFN and IAM datasets may contain images with more than two (handwritten) words, and thus more than two word labels.

3.3.4 CIFAR-100 dataset

CIFAR-100 is a well-known natural images dataset that has heavily been investigated in the computer vision literature. It essentially has $50,000$ 32×32 RGB images for training and $10,000$ 32×32 RGB images for testing. Each image

Figure 2. Visual representation of the six tasks used in this work. A top-down view displays 4 images from distracted driver, text of a movie review in the form of 2D Word2Vec tensor, spectrogram of one spoken word, 4 IAM English handwriting images, 4 IFN Arabic handwriting images, and six images selected from the CIFAR-100. Our model has to learn a total of 6 tasks in three different domains; vision, speech and language.

belongs to one of 100 classes distributed over 600 samples, and each class has 500 and 100 training and testing samples, respectively.

3.3.5 Distracted driver dataset

We use the State Farm Distracted Driver (dDRIVER) dataset that has been collected using 2D dashboard camera [15, 15]. The objective is to classify each driver's behavior that has been categorized to one of ten classes: "SAFE DRIVING", "TEXTING-RIGHT", "TALKING ON THE PHONE-RIGHT", "TEXTING-LEFT", "TALKING ON THE PHONE - LEFT", "OPERATING THE RADIO", "DRINKING", "REACHING BEHIND", "hair and makeup", "talking to passenger". In our setting, the dataset is partitioned into $16,818$ training and $5,605$ testing samples, respectively.

3.4. The sensory layer

We transform all input data samples into a rank 3 tensor, *i.e.* having three channels, 600 width and 300 height. For each type of data, the sensory layer performs specific operations, based on the task and / or domain, to convert the data to a tensor that is fed then to the deep learning model. The details of the sensory layer operations are as follows:

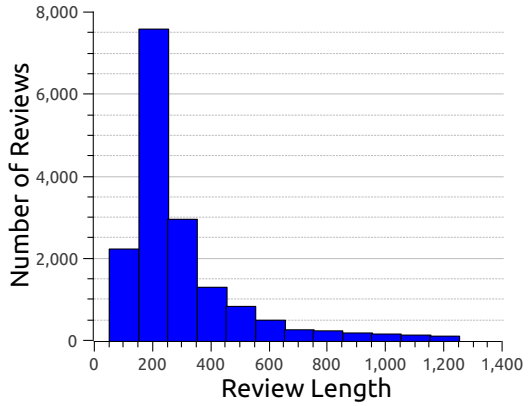

Figure 3. Distribution of review lengths in the IMDB dataset.

- Text sensory layer (IMDB dataset): We first perform cleaning of each text sample (*i.e.* movie review) to remove all non-word symbols. One review before and after cleaning / processing is illustrated in Table 1. The distribution of review lengths in the IMDB dataset is illustrated in Figure 3. In addition, for every review, we convert each word to a numerical vector using Google's pre-trained Word2Vec model [24]. Google's Word2Vec model, which has roughly been trained on 100 billion words from a Google News dataset, has word vectors for a vocabulary of 3 million words and phrases. We use Gensim Python package [30] as an interface to Word2Vec. We use zero padding if the number of words is less than 600 and truncated the review if it has more than 600 words. Each review will result in a rank 2 tensor with resolution of 600×300. Then we repeat the (single) channel 3 times to have a rank 3 tensor of size $600 \times 300 \times 3$. The cleaned movie review shown in Table 1 that we convert using Word2Vec to a rank 3 tensor with width 600 and height 300 is illustrated in the second row of Figure 2.

- Speech sensory layer (SPCHD dataset): We convert each speech signal to its corresponding spectrogram using the Short-Time Fourier Transform [25]. The spectrogram is a visual representation of the spectrum of frequencies of a signal as it varies with time. When applied to an audio signal, spectrograms are sometimes called sonographs, voiceprints, or voicegrams. We pad with zeros each spectrogram in order to have a resolution of 600 width and 300 height. Then we repeat the (single) channel 3 times to have a rank 3 tensor of size $600 \times 300 \times 3$.

- IFN and IAM Handwriting datasets: We convert the pixel intensity values in each image to 1 for the handwriting stroke and 0 for the background. We scale all handwritten images to a height of 120, and the aspect ratio is maintained. We then employ zero padding

Data: Task	#Training samples	#Testing samples	#Classes
IFN: Arabic Handwriting	$24,378$	$8,127$	NA
IAM: English Handwriting	$51,519$	$18,142$	NA
CIFAR-100: Image Classification	$50,000$	$10,000$	100
dDRIVER: Distracted Driver	$16,818$	$5,606$	10
SPCHD: Speech Recognition	$64,721$	$2,489$	30
IMDB: Text Sentiment Analysis	$25,000$	$25,000$	2
Total: Multi-task	$225,963$	$75,915$	NA

Table 2. Data partitioning.

to attain a resolution of 600×300 and channels repeated to three, resulting in a rank 3 tensor of size $600 \times 300 \times 3$.

- Natural images sensory layer for CIFAR-100 and dDRIVER datasets: We scale up the images in the CIFAR-100 dataset, which have the size $32 \times 32 \times 3$, to the height of 300, resulting thus in a tensor of size $300 \times 300 \times 3$. We apply a similar approach to the dDRIVER dataset, as the video has been segmented to frames associated to each action the driver is performing. We scaled-down each image, which originally has the size $640 \times 480 \times 3$, with respect to its height to 300, resulting thus in a tensor of size $400 \times 300 \times 3$. Then, using zero padding, the tensor size of every sample is $600 \times 300 \times 3$.

After these sensory operations, all values in the tensors are then normalized to have zero mean and unit variance; this normalization has become a common practice in computer vision. Moreover, the sensory layer is fixed and does not involve any learning or tuning. Figure 2 provides a visual demonstration of samples selected from each dataset.

3.5. Deep learning model

We train the OneModel to concurrently learn different tasks across various domains. Our model is solely built using one Residual Network with 152 layers (ResNet-152).

3.6. Multiclass accuracy and retrieval metrics

For each testing sample, the model's output layer fires a binary vector denoting the predicted PHOC vector. We use two metrics to measure the performance. First, we calculate the multiclass accuracy that measures the fraction of correct predictions over the testing samples; this metric is dedicated to measure the classification performance. Second, we use the mean Average Precision (mAP), which is the standard evaluation method in retrieval problems, to measure the retrieval performance; this metric is dedicated to measure the word-spotting performance. Similar to other previous works, for example [3, 35], we measure the retrieval performance using two approaches: Query by Example (QbE) [20], and Query by String (QbS) [10]. For QbE, we only use the unique strings in the test set as queries; each sample in the test set is used once as a query to rank the remaining samples in the test set. For QbS, however, we only take the unique samples in the test set and use their PHOC representation as queries. We use the cosine similarity distance to measure the proximity of queries.

4. Results

We build our model using PyTorch framework [26, 29] and we make it publicly available at [1]. We train the model simultaneously using textual sentiment analysis, speech recognition, image classification, action recognition from video, and handwriting word spotting of two different languages / scripts (Arabic and Latin / English). Details related to the sensory input and the output layers, the deep learning model and its hyper-parameters are provided below.

We use ResNet-152 that has been trained with ImageNet [29]. The procedures and the hyper-parameters that we use in the experiments are as follows: SGD, learning-rate ($lr = 0.1$) adapted to ($lr = 0.01$) after 40 epochs, momentum $= 0.9$. We use a binary cross entropy with logits loss function, which combines the binary cross entropy loss and a $Sigmoid$ function and has thus a better numerically stability than a loss function followed by a $Sigmoid$ [29]. Because models trained with a PHOC output layer do not suffer from overfitting [34], we use a maximum of 60 epochs in the training without early stopping criterion. After training is finished, we employ the $Sigmoid$ function on the outputs to bound them between 0 and 1. We use a training batch size equal to 10, as small batch sizes has great generalization advantages [23].

Word hashing modulation is used to resolve collisions between different labels across multiple tasks and domains. In this work, more than 80,000 unique English words and 15,000 unique Arabic words were used to predict multiple tasks across multiple domains. The results in Table 3 show that the OneModel was successful in concurrently learning the 6 different tasks across multiple domain. Furthermore, the results of the OneModel trained on the 6 tasks are on par with the 6 models trained separately to perform a single

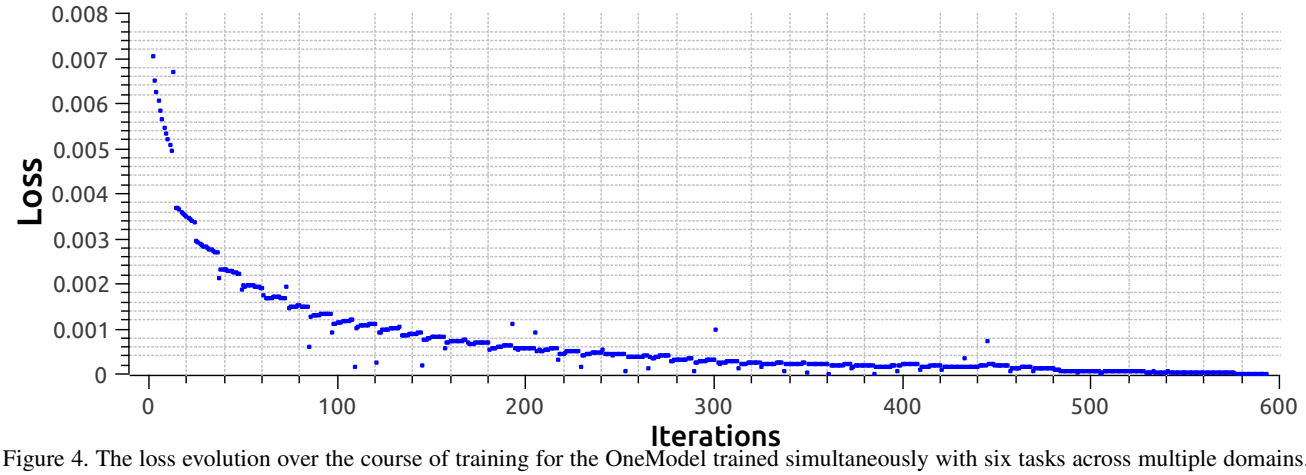

Figure 4. The loss evolution over the course of training for the OneModel trained simultaneously with six tasks across multiple domains. The training loss is demonstrated as a batch_log value such that every ten iterations denotes one epoch.

Data:Task	Joint 6-task	Single task	Single task state-of-the-art
IFN: Arabic Handwriting †	98.1/98.5	99.0/99.0	99.0/99.0 [2]
IAM: English Handwriting †	88.7/89.9	88.5/89.4	94.6/88.3 [34]
CIFAR-100: Image Classification	71.0	74.4	79.3* [8]
dDRIVER: Distracted Driver	93.2	94.5	88.6 [7]
SPCHD: Speech Recognition	96.6	95.9	95.8 [39] ‡
IMDB: Text Sentiment Analysis	81.2	85.3	89.2+ [12]

Table 3. Accuracy [%] comparison of the proposed OneModel trained jointly on 6 tasks or separately on each task. † The retrieval performance of word-spotting is provided in QbS/QbE [%]. ‡ Using a smaller ResNet with only 15 layers. * This accuracy can be reached using data augmentation, which we have not considered in this work. + Such high accuracy can usually been attained using dedicated NLP methods; our approach, however, is novel and is based on converting the text into a rank 3 tensor that can be viewed as an image; it is almost impossible for humans to make inferences from these tensors visually, but clearly, Deep Nets can.

task. Starting from a ResNet-152 trained with ImageNet and after only 10 epochs, the average testing loss decreased to 0.005 and the OneModel reaches an overall accuracy of 64.0%. Notably, the overall chance-level performance of the OneModel, prior to any training, is 0.0%; which is expected due to the undetermined number of classes in the handwriting datasets. The learning loss also decreased from 0.069 to 0.001, which indicates that the OneModel is taking advantage from models trained with ImageNet. The loss evolution over the course of training is shown in Figure 4.

5. Discussion

We show in Figure 5 how the hashing algorithm does a vital role to increase the distance between words / labels, in addition to separating the sensory information. We did the analysis using a character-based word embedding model based on RNN [13], in which similarly spelled words are represented by the neighbouring vectors. For this illustrative example, we use the English stop-words and the three words shown in Figure 5 to build the model. The visualization is based on using Principal Component Analysis projected into 2D space. We show that the hashing algorithm enables the separation between the different tasks / domains when there is a probability of collisions between the words / labels denoting samples across tasks and / or domains.

The sentiment analysis approach we are proposing can be enhanced further as our focus was to get something running within the OneModel context. This can be improved using further NLP techniques at the sensory layer and / or tuning the hyper-parameter of the deep learning model. There are some techniques that can be used in the enhancement; for example, Google's Word2Vec does not recognize British English; *e.g.* the word 'favourite' needs be removed from the textual tensor. This requires adding a spell checker based on American English vs British English to improve the text cleaning. Similarly, our purpose was not to present a model that works best for CIFAR-100 and goes beyond state-of-the-art, *e.g.* as proposed in [8] or other recent regularization methods [43]. All these approaches can be used, in addition to data augmentation to build a better OneModel. With respect to Arabic handwriting word-spotting, our model achieves beyond state-of-the-art performance. Still, for the English word-spotting task,

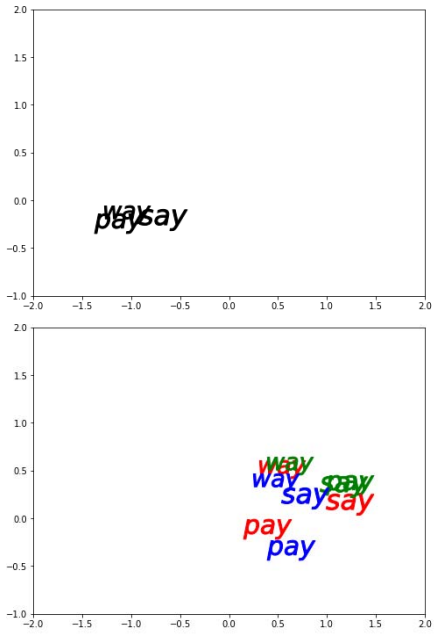

Figure 5. The effect of hashing on words separation and cognition. Top shows the three words 'pay', 'way' and 'say' where one has collision when each word is associated to speech, text/language and vision; we use black to indicate the collision between the different tasks / domains. The bottom figure shoes the effect of using the DJB2 hash algorithm, where red, green, and blue denote speech, vision, and language, respectively. We truncated the hashcodes by taking their modulus with the prime number 1000003, *i.e.* DJB2 mod 1000003. DJB2(payspeech): pay99881; DJB2(payvision): pay712162; DJB2(paylanguage): pay834868; DJB2(wayspeech): way893078; DJB2(wayvision): way505356; DJB2(waylanguage): way623809; DJB2(sayspeech): say296965; DJB2(sayvision): say909246; DJB2(saylanguage): say458699. Best viewed in color.

the OneModel is close to state-of-the-art models. Recent state-of-the-art word-spotting models rely heavily on large data augmentation sets that are obtained via synthetic handwritten generators. We therefore think that incorporating these methods will result in enhancing the overall performance of the OneModel, and not only improving its ability to perform word-spotting. The output layer can be further improved by considering higher PHOC levels or investigating other text embedding techniques. In addition, using Long Short Term Memory networks (LSTMs) or Recurrent Neural Networks (RNNs) on top of the PHOC may provide enhanced reasoning [12].

How does our model fit into how biological vision systems work? In fact, this question has been the major motivation of this work. For example, how does the brains process the word cat? Does the brain picture a cat after reading the word cat? The same argument applies here when one hears the sound of someone saying cat, does the brain picture a cat? Another argument goes in the same and similar manner when one sees a cat or a picture of it, are there any brain activations related a cat as a word / text, or equally related to the sound of someone saying cat? Furthermore, it is well known that the modular regions in the human brain are highly connected, and therefore, it is possible that textual or sound information go to regions dedicated to vision to complete the reasoning. In fact, evidences form functional brain studies supports this [11, 33, 38, 14]. This notion strongly agrees with the OneModel we are proposing in this work, as speech and text are converted into rank 3 tensors that match the visual image space. Regardless of the results we present, the question of how language, vision and speech can be modeled via one deep learning model remains open and further research is needed in this direction.

6. Conclusions

When compared to previous works based on MultiModel deep-learning blocks to build one model, the OneModel approach we are proposing that is based on a single deep-learning model is very successful. To the best of our knowledge, there are no previous works based only on a single model to learn different tasks across multiple domains. We conclude that using an appropriate text output layer that can decode the different tasks across multiple fields from the signals it receive from the deep learning model is provides a compact and efficient learning approach. This can be the bases for further compact models in the future; by using, for example, other textual representation methods for the output layer, improving how the sensory layer processes the input before sending it to the deep learning model, and considering more tasks and domains.

The per-task performance of the OneModel is slightly lower than the best models we trained for every task; still, the OneModel performs fairly well and has a large margin of improvements in the future. The English handwriting words-spotting is an exception, however, as the performance of the OneModel trained on all the tasks is slightly higher than the single model.

There are other problems that remain to be open for future works; for example, using data augmentation, investigating the effect of balancing the data, adding a sensory attention layer, adding more tasks across some other domains, investigating the possibility of the model in zero and a few shot learning, and testing it in domain adaptation problems.

Acknowledgement

This work has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skodowska-Curie grant agreement No. 665919.

References

[1] M. Al-Rawi. MLPHOC: Bi-Script Word Spotting. https://github.com/morawi/MLPHOC/. [Online; accessed: 11-November-2018]. 6

[2] M. Al-Rawi, E. Valveny, , and D. Karatza. Can one deep learning model learn scriptindependent multilingualword-spotting? In *ICDAR*, 2019. 4, 7

[3] J. Almazn, A. Gordo, A. Forns, and E. Valveny. Word spotting and recognition with embedded attributes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(12):2552–2566, 2014. 2, 6

[4] J. Baxter. A bayesian/information theoretic model of learning to learn viamultiple task sampling. *Machine Learning*, 28(1):7–39, July 1997. 2

[5] R. Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997. 2

[6] R. Caruana. *A Dozen Tricks with Multitask Learning*, pages 163–189. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. 2

[7] P. M. Chawan, S. Satardekar, D. Shah, R. Badugu, and A. Pawar. Distracted driver detection and classification. *International Journal of Engineering Research and Applications*, 04 2018. 7

[8] E. D. Cubuk, B. Zoph, D. Mané, V. Vasudevan, and Q. V. Le. Autoaugment: Learning augmentation policies from data. *CoRR*, abs/1805.09501, 2018. 7

[9] J. Deng, N. Ding, Y. Jia, A. Frome, K. Murphy, S. Bengio, Y. Li, H. Neven, and H. Adam. Large-scale object classification using label relation graphs. In *ECCV*, 2014. 2

[10] J. Edwards, Y. W. Teh, D. A. Forsyth, R. Bock, M. Maire, and G. Vesom. Making latin manuscripts searchable using ghmms. In *NIPS*, pages 385–392, 2004. 6

[11] F. Haist, A. W. Song, K. Wild, T. L. Faber, C. A. Popp, and R. D. Morris. Linking sight and sound: fmri evidence of primary auditory cortex activation during visual word recognition. *Brain and Language*, 76(3):340 – 350, 2001. 8

[12] A. Hassan and A. Mahmood. Convolutional recurrent deep learning model for sentence classification. *IEEE Access*, 6:13949–13957, 2018. 7, 8

[13] Intuition-Engineering. State Farm Distracted Driver Detection. https://github.com/IntuitionEngineeringTeam/chars2vec. [Online; accessed 24-April-2019]. 7

[14] X. Jiang, M. A. Chevillet, J. P. Rauschecker, and M. Riesenhuber. Training humans to categorize monkey calls: Auditory feature- and category-selective neural tuning changes. *Neuron*, 98 2:405–416.e4, 2018. 8

[15] Kaggle Competition. State Farm Distracted Driver Detection. https://www.kaggle.com/c/state-farm-distracted-driver-detection, 2016. [Online; accessed 24-Dec-2018]. 5

[16] L. Kaiser, A. N. Gomez, N. Shazeer, A. Vaswani, N. Parmar, L. Jones, and J. Uszkoreit. One model to learn them all. *CoRR*, abs/1706.05137, 2017. 1, 2

[17] P. Krishnan, K. Dutta, and C. V. Jawahar. Word spotting and recognition using deep embedding. *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*, pages 1–6, 2018. 4

[18] M. Long and J. Wang. Learning multiple tasks with deep relationship networks. *CoRR*, abs/1506.02117, 2015. 2

[19] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 142–150, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. 4

[20] R. Manmatha, C. Han, and E. M. Riseman. Word spotting: A new approach to indexing handwriting. In *CVPR*, 1996. 6

[21] U.-V. Marti and H. Bunke. A full English sentence database for off-line handwriting recognition. In *In Proc. Int. Conf. on Document Analysis and Recognition*, pages 705–708, 1999. 4

[22] U.-V. Marti and H. Bunke. The IAM-database: an English sentence database for offline handwriting recognition. *International Journal on Document Analysis and Recognition*, 5:39–46, 2002. 4

[23] D. Masters and C. Luschi. Revisiting small batch training for deep neural networks. *CoRR*, abs/1804.07612, 2018. 6

[24] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013. 5

[25] A. V. Oppenheim. Speech spectrograms using the fast fourier transform. *IEEE Spectrum*, 7(8):57–62, 1970. 5

[26] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017. 6

[27] M. Pechwitz, S. S. Maddouri, V. Mrgner, N. Ellouze, and H. Amiri. Ifn/enit - database of handwritten arabic words. In *In Proc. of CIFED 2002*, pages 129–136, 2002. 4

[28] M. Pechwitz, S. S. Maddouri, V. Mrgner, N. Ellouze, and H. Amiri. IFN/ENIT Arabic handwriting dataset. http://www.ifnenit.com/, 2002. [Online; accessed: 20-may-2018]. 4

[29] PyTorch. Tensors and Dynamic neural networks in Python with strong GPU acceleration. https://pytorch.org/, 2018. [Online; accessed 14-Dec-2018]. 6

[30] R. Řehůřek and P. Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. 5

[31] V. Roussev, G. G. Richard, and L. Marziale. Multi-resolution similarity hashing. 2007. 3

[32] S. Ruder. An overview of multi-task learning in deep neural networks. *CoRR*, abs/1706.05098, 2017. 1, 2

[33] J. G. Rueckl, P. M. Paz-Alonso, P. J. Molfese, W. S. Kuo, A. S. Bick, S. J. Frost, R. Hancock, D. H. Wu, W. E. Mencl, J. A. Duñabeitia, J.-R. Lee, M. Oliver, J. D. Zevin, F. Hoeft, M. Carreiras, O. J. L. Tzeng, K. R. Pugh, and R. Frost. Universal brain signature of proficient reading: Evidence from four contrasting languages. *Proceedings of the National Academy of Sciences of the United States of America*, 112 50:15510–5, 2015. 8

[34] E. Rusakov, S. Sudholt, F. Wolf, and G. Fink. Explorring architectures for cnn-based word spotting. *CoRR*, abs/1806.10866, 2018. 4, 6, 7

[35] S. Sudholt and G. A. Fink. PHOCNet: A deep convolutional neural network for word spotting in handwritten documents. In *Frontiers in Handwriting Recognition (ICFHR), 2016 15th International Conference on*, pages 277–282. IEEE, 2016. 4, 6

[36] S. Sudholt and G. A. Fink. Evaluating word string embeddings and loss functions for cnn-based word spotting. In *ICDAR*, pages 493–498. IEEE, 2017. 4

[37] S. Sudholt and G. A. Fink. Attribute CNNs for word spotting in handwritten documents. *International Journal on Document Analysis and Recognition (IJDAR)*, 21:199–218, 2018. 4

[38] S. Sutherland. When we read, we recognize words as pictures and hear them spoken aloud. *Scientific American: Mind*, 2015. 8

[39] R. Tang and J. Lin. Deep residual learning for small-footprint keyword spotting. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5484–5488, 2018. 7

[40] P. Warden. Speech commands: A public dataset for single-word speech recognition. http://download.tensorflow.org/data/speech_commands_v0.01.tar.gz, 2017. [Online; accessed: 20-12-2018]. 4

[41] P. Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *CoRR*, abs/1804.03209, 2018. 4

[42] Y. Xia, X. Tan, F. Tian, T. Qin, N. Yu, and T.-Y. Liu. Model-level dual learning. In *ICML*, 2018. 2

[43] Y. Yamada, M. Iwamura, and K. Kise. Shakedrop regularization. *CoRR*, abs/1802.02375, 2018. 7