Correlational Neural Networks

Sarath Chandar[1, 3, 4], **Mitesh M Khapra**[1], **Hugo Larochelle**[2], **Balaraman Ravindran**[3]

[1]IBM Research India.

[2]University of Sherbrooke.

[3]Indian Institute of Technology Madras.

[4]Corresponding author. email: apsarathchandar@gmail.com

Keywords: Representation Learning, Deep Learning, Transfer Learning, Neural Networks, Autoencoders, Common Representations.

Abstract

Common Representation Learning (CRL), wherein different descriptions (or views) of the data are embedded in a common subspace, is receiving a lot of attention recently. Two popular paradigms here are Canonical Correlation Analysis (CCA) based approaches and Autoencoder (AE) based approaches. CCA based approaches learn a joint representation by maximizing correlation of the views when projected to the common subspace. AE based methods learn a common representation by minimizing the error of reconstructing the two views. Each of these approaches has its own advantages and disadvantages. For example, while CCA based approaches outperform AE based approaches for the task of transfer learning, they are not as scalable as the latter. In this work we propose an AE based approach called Correlational Neural Network (CorrNet), that explicitly maximizes correlation among the views when projected to the common subspace. Through a series of experiments, we demonstrate that the proposed

CorrNet is better than the above mentioned approaches with respect to its ability to learn correlated common representations. Further, we employ CorrNet for two cross language tasks and show that the representations learned using CorrNet perform better than the ones learned using other state of the art approaches.

1 Introduction

In several real world applications, the data contains more than one view. For example, a movie clip has three views (of different modalities) : audio, video and text/subtitles. However, all the views may not always be available. For example, for many movie clips, audio and video may be available but subtitles may not be available. Recently there has been a lot of interest in learning a common representation for multiple views of the data which can be useful in several downstream applications when some of the views are missing. We consider three applications to motivate the importance of learning common representations: (i) reconstruction of a missing view, (ii) transfer learning and (iii) matching corresponding items across views.

In the first application, the learned common representations can be used to train a model to reconstruct all the views of the data (akin to autoencoders reconstructing the input view from a hidden representation). Such a model would allow us to reconstruct the subtitles even when only audio/video is available. Now, as an example of transfer learning, consider the case where a profanity detector trained on movie subtitles needs to detect profanities in a movie clip for which only video is available. If a common representation is available for the different views, then such detectors/classifiers can be trained by computing this common representation from the relevant view (subtitles, in the above example). At test time, a common representation can again be computed from the available view (video, in this case) and this representation can be fed to the trained model for prediction. Finally, consider the case where items from one view (say, names written using the script of one language) need to be matched to their corresponding items from another view (names written using the script of another language). One way of doing this is to project items from the two views to a common subspace such that the common representations of corresponding items from the two views are correlated. We

can then match items across views based on the correlation between their projections.

Having motivated the importance of Common Representation Learning (CRL), we now formally define this task. Consider some data $\mathcal{Z} = \{\mathbf{z}_i\}_{i=1}^N$ which has two views: X and Y. Each data point $\mathbf{z}_i$ can be represented as a concatenation of these two views : $\mathbf{z}_i = (\mathbf{x}_i, \mathbf{y}_i)$, where $\mathbf{x}_i \in \mathbb{R}^{d_1}$ and $\mathbf{y}_i \in \mathbb{R}^{d_2}$. In this work, we are interested in learning two functions, h_X and h_Y, such that $h_X(\mathbf{x}_i) \in \mathbb{R}^k$ and $h_Y(\mathbf{y}_i) \in \mathbb{R}^k$ are projections of $\mathbf{x}_i$ and $\mathbf{y}_i$ respectively in a common subspace ($\mathbb{R}^k$) such that for a given pair $\mathbf{x}_i$, $\mathbf{y}_i$:

1. $h_X(\mathbf{x}_i)$ and $h_Y(\mathbf{y}_i)$ should be highly correlated.

2. It should be possible to reconstruct $\mathbf{y}_i$ from $\mathbf{x}_i$ (through $h_X(\mathbf{x}_i)$) and vice versa.

Canonical Correlation Analysis (CCA) (Hotelling, 1936) is a commonly used tool for learning such common representations for two-view data (Udupa and Khapra, 2010; Dhillon et al., 2011). By definition, CCA aims to produce correlated common representations but, it suffers from some drawbacks. First, it is not easily scalable to very large datasets. Of course, there are some approaches which try to make CCA scalable (for example, (Lu and Foster, 2014)), but such scalability comes at the cost of performance. Further, CCA does not have any reconstruction capabilities, *i.e.*, it cannot be used to reconstruct one view from the other. Finally, CCA cannot benefit from additional non-parallel, single-view data. This puts it at a severe disadvantage in several real world situations, where in addition to some parallel two-view data, abundant single view data is available for one or both views.

Recently, Multimodal Autoencoders (MAEs) (Ngiam et al., 2011) have been proposed to learn a common representation for two views/modalities. The idea in MAE is to train an autoencoder to perform two kinds of reconstruction. Given any one view, the model learns both self-reconstruction and cross-reconstruction (reconstruction of the other view). This makes the representations learnt to be predictive of each other. However, it should be noticed that the MAE does not get any explicit learning signal encouraging it to share the capacity of its common hidden layer between the views. In other words, it could develop units whose activation is dominated by a single view. This makes the MAE not suitable for transfer learning, since the views are not guaranteed to be projected to a common subspace. This is indeed verified by the results reported in

(Ngiam et al., 2011) where they show that CCA performs better than deep MAE for the task of transfer learning.

These two approaches have complementary characteristics. On one hand, we have CCA and its variants which aim to produce correlated common representations but lack reconstruction capabilities. On the other hand, we have MAE which aims to do self-reconstruction and cross-reconstruction but does not guarantee correlated common representations. In this paper, we propose Correlational Neural Network (CorrNet) as a method for learning common representations which combines the advantages of the two approaches described above. The main characteristics of the proposed method can be summarized as follows:

- It allows for self/cross reconstruction. Thus, unlike CCA (and like MAE) it has predictive capabilities. This can be useful in applications where a missing view needs to be reconstructed from an existing view.

- Unlike MAE (and like CCA) the training objective used in CorrNet ensures that the common representations of the two views are correlated. This is particularly useful in applications where we need to match items from one view to their corresponding items in the other view.

- CorrNet can be trained using Gradient Descent based optimization methods. Particularly, when dealing with large high dimensional data, one can use Stochastic Gradient Descent with mini-batches. Thus, unlike CCA (and like MAE) it is easy to scale CorrNet.

- The procedure used for training CorrNet can be easily modified to benefit from additional single view data. This makes CorrNet useful in many real world applications where additional single view data is available.

We evaluate CorrNet using three different experimental setups. First, we use the MNIST hand-written digit recognition dataset to compare CorrNet with other state of the art CRL approaches. In particular, we evaluate its (i) ability to self/cross reconstruct (ii) ability to produce correlated common representations and (iii) usefulness in transfer learning. In this setup, we use the left and right halves of the digit images as two

views. Next, we use CorrNet for a transfer learning task where the two views of data come from two different languages. Specifically, we use CorrNet to project parallel documents in two languages to a common subspace. We then employ these common representations for the task of cross language document classification (transfer learning) and show that they perform better than the representations learned using other state of the art approaches. Finally, we use CorrNet for the task of transliteration equivalence where the aim is to match a name written using the script of one language (first view) to the same name written using the script of another language (second view). Here again, we demonstrate that with its ability to produce better correlated common representations, CorrNet performs better than CCA and MAE.

The remainder of this paper is organized as follows. In section 2 we describe the architecture of CorrNet and outline a training procedure for learning its parameters. In section 3 we propose a deep variant of CorrNet. In section 4 we briefly discuss some related models for learning common representations. In section 5 we present experiments to analyze the characteristics of CorrNet and compare it with CCA, KCCA and MAE. In section 6 we empirically compare Deep CorrNet with some other deep CRL methods. In sections 7 and 8 we report results obtained by using CorrNet for the tasks of cross language document classification and transliteration equivalence detection respectively. Finally, we present concluding remarks in section 9 and highlight possible future work.

2 Correlational Neural Network

As described earlier, our aim is to learn a common representation from two views of the same data such that: (i) any single view can be reconstructed from the common representation, (ii) a single view can be predicted from the representation of another view and (iii) like CCA, the representations learned for the two views are correlated. The first goal above can be achieved by a conventional autoencoder. The first and second can be achieved together by a Multimodal autoencoder but it is not guaranteed to project the two views to a common subspace. We propose a variant of autoencoders which can work with two views of the data, while being explicitly trained to achieve all

the above goals. In the following sub-sections, we describe our model and the training procedure.

2.1 Model

We start by proposing a neural network architecture which contains three layers: an input layer, a hidden layer and an output layer. Just as in a conventional single view autoencoder, the input and output layers have the same number of units, whereas the hidden layer can have a different number of units. For illustration, we consider a two-view input $\mathbf{z} = (\mathbf{x}, \mathbf{y})$. For all the discussions, $[\mathbf{x}, \mathbf{y}]$ denotes a concatenated vector of size $d_1 + d_2$.

Given $\mathbf{z} = (\mathbf{x}, \mathbf{y})$, the hidden layer computes an encoded representation as follows:

$$h(\mathbf{z}) = f(\mathbf{W}\mathbf{x} + \mathbf{V}\mathbf{y} + \mathbf{b})$$

where $\mathbf{W}$ is a $k \times d_1$ projection matrix, $\mathbf{V}$ is a $k \times d_2$ projection matrix and $\mathbf{b}$ is a $k \times 1$ bias vector. Function f can be any non-linear activation function, for example *sigmoid* or *tanh*. The output layer then tries to reconstruct $\mathbf{z}$ from this hidden representation by computing

$$\mathbf{z}' = g([\mathbf{W}'h(\mathbf{z}), \mathbf{V}'h(\mathbf{z})] + \mathbf{b}')$$

where $\mathbf{W}'$ is a $d_1 \times k$ reconstruction matrix, $\mathbf{V}'$ is a $d_2 \times k$ reconstruction matrix and $\mathbf{b}'$ is a $(d_1 + d_2) \times 1$ output bias vector. Vector $\mathbf{z}'$ is the reconstruction of $\mathbf{z}$. Function g can be any activation function. This architecture is illustrated in Figure 1. The parameters of the model are $\theta = \{\mathbf{W}, \mathbf{V}, \mathbf{W}', \mathbf{V}', \mathbf{b}, \mathbf{b}'\}$. In the next sub-section we outline a procedure for learning these parameters.

2.2 Training

Restating our goals more formally, given a two-view data $\mathcal{Z} = \{(\mathbf{z}_i)\}_{i=1}^N = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$, for each instance, $(\mathbf{x}_i, \mathbf{y}_i)$, we would like to:

- Minimize the self-reconstruction error, *i.e.*, minimize the error in reconstructing $\mathbf{x}_i$ from $\mathbf{x}_i$ and $\mathbf{y}_i$ from $\mathbf{y}_i$.

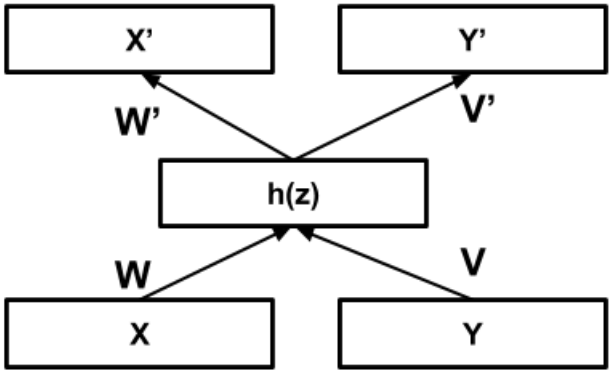

Figure 1: Correlational Neural Network

- Minimize the cross-reconstruction error, *i.e.*, minimize the error in reconstructing $\mathbf{x}_i$ from $\mathbf{y}_i$ and $\mathbf{y}_i$ from $\mathbf{x}_i$.
- Maximize the correlation between the hidden representations of both views.

We achieved this by finding the parameters $\theta = \{\mathbf{W}, \mathbf{V}, \mathbf{W}', \mathbf{V}', \mathbf{b}, \mathbf{b}'\}$ which minimize the following objective function:

$$\mathcal{J}_{\mathcal{Z}}(\theta) = \sum_{i=1}^{N}(L(\mathbf{z}_i, g(h(\mathbf{z}_i))) + L(\mathbf{z}_i, g(h(\mathbf{x}_i))) + L(\mathbf{z}_i, g(h(\mathbf{y}_i)))) - \lambda \operatorname{corr}(h(X), h(Y))$$

$$\operatorname{corr}(h(X), h(Y)) = \frac{\sum_{i=1}^{N}(h(\mathbf{x}_i) - \overline{h(X)})(h(\mathbf{y}_i) - \overline{h(Y)})}{\sqrt{\sum_{i=1}^{N}(h(\mathbf{x}_i) - \overline{h(X)})^2 \sum_{i=1}^{N}(h(\mathbf{y}_i) - \overline{h(Y)})^2}}$$

where L is the reconstruction error, λ is the scaling parameter to scale the fourth term with respect to the remaining three terms, $\overline{h(X)}$ is the mean vector for the hidden representations of the first view and $\overline{h(Y)}$ is the mean vector for the hidden representations of the second view. If all dimensions in the input data take binary values then we use cross-entropy as the reconstruction error otherwise we use squared error loss as the reconstruction error. For simplicity, we use the shorthands $h(\mathbf{x}_i)$ and $h(\mathbf{y}_i)$ to note the representations $h((\mathbf{x}_i, 0))$ and $h((0, \mathbf{y}_i))$ that are based only on a single view[1]. The correlation term in the objective function is calculated considering the hidden representation as a random vector.

[1] They represent the generic functions h_X and h_Y mentioned in the introduction.

In words, the objective function decomposes as follows. The first term is the usual autoencoder objective function which helps in learning meaningful hidden representations. The second term ensures that both views can be predicted from the shared representation of the first view alone. The third term ensures that both views can be predicted from the shared representation of the second view alone. The fourth term interacts with the other objectives to make sure that the hidden representations are highly correlated, so as to encourage the hidden units of the representation to be shared between views.

We can use stochastic gradient descent (SGD) to find the optimal parameters. For all our experiments, we used mini-batch SGD. The fourth term in the objective function is then approximated based on the statistics of a minibatch. Approximating second order statistics using minibatches for training was also used successfully in the batch normalization training method of Ioffe and Szegedy (2015).

The model has three hyperparameters: (i) the number of units in its hidden layer, (ii) λ and (iii) the SGD learning rate. The first hyperparameter is dependent on the specific task at hand and can be tuned using a validation set (exactly as is done by other competing algorithms). The second hyperparameter is only to ensure that the correlation term in the objective function has the same range as the reconstruction errors. This is again easy to approximate based on the given data. The final hyperparameter, the learning rate is common for all neural network based approaches.

Once the parameters are learned, we can use the CorrNet to compute representations of views that can potentially generalize across views. Specifically, given a new data instance for which only one view is available, we can compute its corresponding representation ($h(\mathbf{x})$ if $\mathbf{x}$ is observed or $h(\mathbf{y})$ if $\mathbf{y}$ is observed) and use it as the new data representation.

2.3 Using additional single view data

In practice, it is often the case that we have abundant single view data and comparatively little two-view data. For example, in the context of text documents from two languages (X and Y), typically the amount of monolingual (single view) data available in each language is much larger than parallel (two-view) data available between X and

Y. Given the abundance of such single view data, it is desirable to exploit it in order to improve the learned representation. CorrNet can achieve this, by using the single view data to improve the self-reconstruction error as explained below.

Consider the case where, in addition to the data $\mathcal{Z} = \{(\mathbf{z}_i)\}_{i=1}^{N} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N}$, we also have access to the single view data $\mathcal{X} = \{(\mathbf{x}_i)\}_{i=N+1}^{N_1}$ and $\mathcal{Y} = \{(\mathbf{y}_i)\}_{i=N+1}^{N_2}$. Now, during training, in addition to using $\mathcal{Z}$ as explained before, we also use $\mathcal{X}$ and $\mathcal{Y}$ by suitably modifying the objective function so that it matches that of a conventional autoencoder. Specifically, when we have only $\mathbf{x}_i$, then we could try to minimize

$$\mathcal{J}_{\mathcal{X}}(\theta) = \sum_{i=N+1}^{N_1} L(\mathbf{x}_i, g(h(\mathbf{x}_i)))$$

and similarly for $\mathbf{y}_i$.

In all our experiments, when we have access to all three types of data (*i.e.*, $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$), we construct 3 sets of mini-batches by sampling data from $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$ respectively. We then feed these mini-batches in random order to the model and perform a gradient update based on the corresponding objective function.

3 Deep Correlational Neural Networks

An obvious extension for CorrNets is to allow for multiple hidden layers. The main motivation for having such Deep Correlational Neural Networks is that a better correlation between the views of the data might be achievable by more non-linear representations.

We use the following procedure to train a Deep CorrNet.

1. Train a shallow CorrNet with the given data (see step-1 in Figure 2). At the end of this step, we have learned the parameters $\mathbf{W}$, $\mathbf{V}$ and $\mathbf{b}$.

2. Modify the CorrNet model such that the first input view connects to a hidden layer using weights $\mathbf{W}$ and bias $\mathbf{b}$. Similarly connect the second view to a hidden layer using weights $\mathbf{V}$ and bias $\mathbf{b}$. We have now decoupled the common hidden layer for each view (see step-2 in Figure 2).

3. Add a new common hidden layer which takes its input from the hidden layers

created at step 2. We now have a CorrNet which is one layer deeper (see step-3 in Figure 2).

4. Train the new Deep CorrNet on the same data.

5. Repeat steps 2, 3 and 4, for as many hidden layers as required.

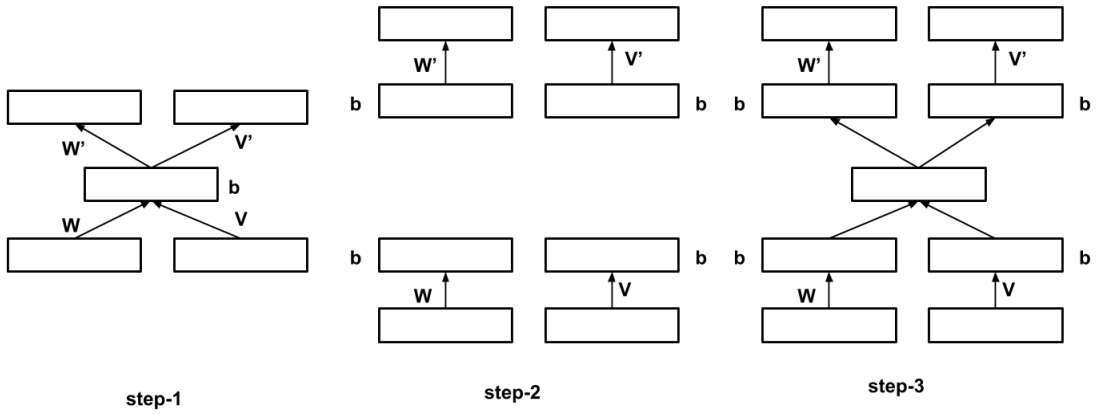

Figure 2: Stacking CorrNet to create Deep Correlartional Neural Network.

We would like to point out that we could have followed the procedure described in Chandar et al. (2013) for training Deep CorrNet. In Chandar et al. (2013), they learn deep representation for each view separately and use it along with a shallow CorrNet to learn a common representation. However, feeding non-linear deep representations to a shallow CorrNet makes it harder to train the CorrNet. Also, we chose not to use the deep training procedure described in Ngiam et al. (2011) since the objective function used by them during pre-training and training is different. Specifically, during pre-training the objective is to minimize self reconstruction error whereas during training the objective is to minimize both self and cross reconstruction error. In contrast, in the stacking procedure outlined above, the objectives during training and pre-training are aligned.

4 Related Models

In this section, we briefly describe other neural network based common representation learning models and explain how CorrNet differs from them.

Hsieh (2000) is one of the earliest Neural Network based model for nonlinear CCA. This method uses three feedforward neural networks. The first neural network is a double-barreled architecture where two networks project the views to a single unit such that the projections are maximally correlated. This network is first trained to maximize the correlation. Then the inverse mapping for each view is learnt from the corresponding canonical covariate representation by minimizing the reconstruction error. There are clear differences between this Neural CCA model and CorrNet. First, CorrNet is a single neural network which is trained with a single objective function while Neural CCA has three networks trained with different objective functions. Second, Neural CCA does only correlation maximization and self-reconstruction, whereas CorrNet does correlation maximization, self-reconstruction and cross-reconstruction, all at the same time.

Multimodal Autoencoder (MAE) (Ngiam et al., 2011) is another Neural Network based CRL approach. Even though the architecture of MAE is similar to that of CorrNet there are clear differences in the training procedure used by the two. Firstly, MAE only aims to minimize the following three errors: (i) error in reconstructing z_i from x_i (E_1), (ii) error in reconstructing z_i from y_i (E_2) and (iii) error in reconstructing z_i from z_i (E_3). More specifically, unlike the fourth term in our objective function, the objective function used by MAE does not contain any term which forces the network to learn correlated common representations. Secondly, there is a difference in the manner in which these terms are considered during training. Unlike CorrNet, MAE only considers one of the above terms at a time. In other words, given an instance $z_i = (x_i, y_i)$ it first tries to minimize E_1 and updates the parameters accordingly. It then tries to minimize E_2 followed by E_3. Empirically, we observed that a training procedure which considers all three loss terms together performs better than the one which considers them separately.

Deep Canonical Correlation Analysis (DCCA) (Andrew et al., 2013) is a recently proposed Neural Network approach for CCA. DCCA employs two deep networks, one per view. The model is trained in such a way that the final layer projections of the data

in both the views are maximally correlated. DCCA maximizes only correlation whereas CorrNet maximizes both, correlation and reconstruction ability.

5 Analysis of Correlational Neural Networks

In this section, we perform a set of experiments to compare CorrNet, CCA (Hotelling, 1936), Kernel CCA (KCCA) (Akaho, 2001) and MAE (Ngiam et al., 2011) based on:

- ability to reconstruct a view from itself
- ability to reconstruct one view given the other
- ability to learn correlated common representations for the two views
- usefulness of the learned common representations in transfer learning.

For CCA, we used a C++ library called *dlib* (King, 2009). For KCCA, we used an implementation provided by the authors of (Arora and Livescu, 2012). We implemented CorrNet and MAE using Theano (Bergstra et al., 2010).

5.1 Data Description

We used the standard MNIST handwritten digits image dataset for all our experiments. This data consists of 60,000 train images and 10,000 test images. Each image is a 28 * 28 matrix of pixels; each pixel representing one of 256 grayscale values. We treated the left half of the image as one view and the right half of the image as another image. Thus each view contains $14 * 28 = 392$ dimensions. We split the train images into two sets. The first set contains 50,000 images and is used for training. The second set contains 10,000 images and is used as a validation set for tuning the hyper-parameters of the four models described above.

5.2 Performance of Self and Cross Reconstruction

Among the four models listed above, only CorrNets and MAE have the ability to construct a view from itself as well as from the other view. So, in this sub-section, we

consider only these two models. Table 1 shows the Mean Squared Errors (MSEs) for self and cross reconstruction when the left half of the image is used as input.

Model	**MSE for self reconstruction**	**MSE for cross reconstruction**
CorrNet	3.6	4.3
MAE	**2.1**	**4.2**

Table 1: Mean Squared Error for CorrNet and MAE for self reconstruction and cross reconstruction

The above table suggests that CorrNet has a higher self reconstruction error and almost the same cross reconstruction error as that of MAE. This is because unlike MAE, in CorrNet, the emphasis is on maximizing the correlation between the common representations of the two views. This goal captured by the fourth term in the objective function obviously interferes with the goal of self reconstruction. As we will see in the next sub-section, the embeddings learnt by CorrNet for the two views are better correlated even though the self-reconstruction error is sacrificed in the process.

Figure 3 shows the reconstruction of the right half from the left half for a few sample images. The figure reiterates our point that both CorrNet and MAE are equally good at cross reconstruction.

Figure 3: Reconstruction of right half of the image given the left half. First block shows the original images, second block shows images where the right half is reconstructed by CorrNet and the third block shows images where the right half is reconstructed by MAE.

5.3 Correlation between representations of two views

As mentioned above, in CorrNet we emphasize on learning highly correlated representations for the two views. To show that this is indeed the case, we follow (Andrew et al., 2013) and calculate the total/sum correlation captured in the 50 dimensions of the common representations learnt by the four models described above. The training, validation and test sets used for this experiment were as described in section 5.1. The results are reported in Table 2.

Model	**Sum Correlation**
CCA	17.05
KCCA	30.58
MAE	24.40
CorrNet	**45.47**

Table 2: Sum/Total correlation captured in the 50 dimensions of the common representations learned by different models using MNIST data.

The total correlation captured in the 50 dimensions learnt by CorrNet is clearly better than that of the other models.

Next, we check whether this is indeed the case when we change the number of dimensions. For this, we varied the number of dimensions from 5 to 80 and plotted the sum correlation for each model (see Figure 4). For all the models, we tuned the hyper-parameters for $dim = 50$ and used the same hyper-parameters for all dimensions.

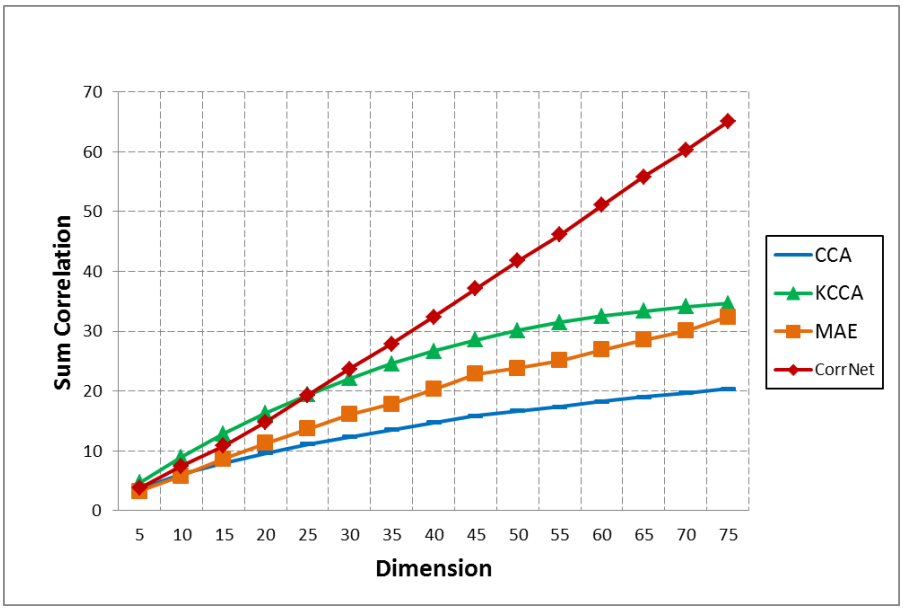

Figure 4: Sum/Total correlation as a function of the number of dimensions in the common representations learned by different models using MNIST data.

Again, we see that CorrNet clearly outperforms the other models. CorrNet thus achieves its primary goal of producing correlated embeddings with the aim of assisting transfer learning.

5.4 Transfer Learning across views

To demonstrate transfer learning, we take the task of predicting digits from only one half of the image. We first learn a common representation for the two views using 50,000 images from the MNIST training data. For each training instance, we take only one half of the image and compute its 50 dimensional common representation using one of the models described above. We then train a classifier using this representation. For each test instance, we consider only the other half of the image and compute its common representation. We then feed this representation to the classifier for prediction. We use the linear SVM implementation provided by (Pedregosa et al., 2011) as the classifier for all our experiments and do 5-fold cross validation using 10000 test images. We report accuracy for two settings (i) Left to Right (training on left view, testing on right view) and (ii) Right to Left (training on right view, testing on left view).

Model	Left to Right	Right to Left
CCA	65.73	65.44
KCCA	68.1	75.71
MAE	64.14	68.88
CorrNet	**77.05**	**78.81**

Table 3: Transfer learning accuracy using the representations learned using different models on the MNIST dataset.

Once again, we see that CorrNet performs significantly better than the other models. To verify that this holds even when we decrease the data for learning common representation to 10000 images. The results as reported in Table 4 show that even with less data, CorrNet perform betters than other models.

Model	Left to Right	Right to Left
CCA	66.13	66.71
KCCA	70.68	70.83
MAE	68.69	72.54
CorrNet	**76.6**	**79.51**

Table 4: Transfer learning accuracy using the representations learned using different models trained with 10000 instances from the MNIST dataset.

5.5 Analysis of Loss Terms

The objective function defined in Section 2.2 has the following four terms:

- $L_1 = \sum_{i=1}^{N} L(\mathbf{z}_i, g(h(\mathbf{z}_i))$
- $L_2 = \sum_{i=1}^{N} L(\mathbf{z}_i, g(h(\mathbf{x}_i))$
- $L_3 = \sum_{i=1}^{N} L(\mathbf{z}_i, g(h(\mathbf{y}_i))$
- $L_4 = \lambda \operatorname{corr}(h(X), h(Y))$

In this section, we analyze the importance of each of these terms in the loss function. For this, during training, we consider different loss functions which contain different

combinations of these terms. In addition, we consider two more loss terms for our analysis.

- $L_5 = \sum_{i=1}^{N} L(\mathbf{y}_i, g(h(\mathbf{x}_i))$
- $L_6 = \sum_{i=1}^{N} L(\mathbf{x}_i, g(h(\mathbf{y}_i))$

where $L5$ and $L6$ essentially capture the loss in reconstructing only one view (say, $\mathbf{x}_i$) from the other view ($\mathbf{y}_i$).

For this, we first learn common representations using different loss functions as listed in the first column of Table 5. We then repeated the transfer learning experiments using common representations learned from each of these models. For example, the sixth row in the table shows the results when the following loss function is used for learning the common representations.

$$\mathcal{J}_{\mathcal{Z}}(\theta) = L_1 + L_2 + L_3 + L_4$$

which is the same as that used in CorrNet.

Loss function used for training	**Left to Right**	**Right to Left**
L_1	24.59	22.56
$L_1 + L_4$	65.9	67.54
$L_2 + L_3$	71.54	75
$L_2 + L_3 + L_4$	76.54	**80.57**
$L_1 + L_2 + L_3$	67.82	72.13
$L_1 + L_2 + L_3 + L_4$	**77.05**	78.81
$L_5 + L_6$	35.62	32.26
$L_5 + L_6 + L_4$	62.05	63.49

Table 5: Comparison of the performance of transfer learning with representations learned using different loss functions.

Each even numbered row in the table reports the performance when the correlation term (L_4) was used in addition to the other terms in the row immediately before it. A

pair-wise comparison of the numbers in each even numbered row with the row immediately above it suggests that correlation term (L_4) in the loss function clearly produces representations which lead to better transfer learning.

6 Experiments using Deep Correlational Neural Network

In this section, we evaluate the performance of the deep extension of CorrNet. Having already compared with MAE in the previous section, we focus our evaluation here on a comparison with DCCA (Andrew et al., 2013). All the models were trained using 10000 images from the MNIST training dataset and we computed the sum correlation and transfer learning accuracy for each of these models. For transfer learning, we use the linear SVM implementation provided by (Pedregosa et al., 2011) for all our experiments and do 5-fold cross validation using 10000 test images from MNIST data. We report results for two settings (i) Left to Right (training on left view, testing on right view) and (ii) Right to Left (training on right view, testing on left view). These results are summarized in Table 6. In this Table, model-x-y means a model with x units in the first hidden layer and y units in second hidden layer. For example, CorrNet-500-300-50 is a Deep CorrNet with three hidden layers containing 500, 300 and 50 units respectively. The third layer containing 50 units is used as the common representation.

Model	Sum Correlation	Left to Right	Right to Left
CorrNet-500-50	**47.21**	77.68	77.95
DCCA-500-50	33.00	66.41	64.65
CorrNet-500-300-50	45.634	**80.46**	**80.47**
DCCA-500-500-50	33.77	70.06	72.43

Table 6: Comparison of sum correlation and transfer learning performance of different deep models

Both the Deep CorrNets (CorrNet-500-50 and CorrNet-500-300-50) clearly perform better than the corresponding DCCA. We notice that for both the transfer learning tasks, the 3-layered CorrNet (CorrNet-500-300-50) performs better than the 2-layered Corr-

Net (CorrNet-500-50) but the sum correlation of the 2-layered CorrNet is better than that of the 3-layered CorrNet.

7 Cross Language Document Classificationn

In this section, we will learn bilingual word representations using CorrNet and use these representations for the task of cross language document classification. We experiment with three language pairs and show that our approach achieves state-of-the-art performance.

Before we discuss bilingual word representations let us consider the task of learning word representations for a single language. Consider a language X containing d words in its vocabulary. We represent a sentence in this language using a binary bag-of-words representation $\mathbf{x}$. Specifically, each dimension x_i is set to 1 if the i^{th} vocabulary word is present in the sentence, 0 otherwise. We wish to learn a k-dimensional vectorial representation of each word in the vocabulary from a training set of sentence bags-of-words $\{\mathbf{x}_i\}_{i=1}^{N}$.

We propose to achieve this by using a CorrNet which works with only a single view of the data (see section 2.3). Effectively, one can view a CorrNet as encoding an input bag-of-words $\mathbf{x}$ as the sum of the columns in $\mathbf{W}$ corresponding to the words that are present in $\mathbf{x}$, followed by a non-linearity. Thus, we can view $\mathbf{W}$ as a matrix whose columns act as vector representations (embeddings) for each word.

Let's now assume that for each sentence bag-of-words $\mathbf{x}$ in some source language X, we have an associated bag-of-words $\mathbf{y}$ for this sentence translated in some target language Y by a human expert. Assuming we have a training set of such $(\mathbf{x}, \mathbf{y})$ pairs, we'd like to learn representations in both languages that are aligned, such that pairs of translated words have similar representations. The CorrNet can allow us to achieve this. Indeed, it will effectively learn word representations (the columns of $\mathbf{W}$ and $\mathbf{V}$) that are not only informative about the words present in sentences of each language, but will also ensure that the representations' space is aligned between language, as required by the cross-view reconstruction terms and the correlation term.

Note that, since the binary bags-of-words are very high-dimensional (the dimension-

ality corresponds to the size of the vocabulary, which is typically large), reconstructing each binary bag-of-word will be slow. Since we will later be training on millions of sentences, training on each individual sentence bag-of-words will be expensive. Thus, we propose a simple trick, which exploits the bag-of-words structure of the input. Assuming we are performing mini-batch training (where a mini-batch contains a list of the bags-of-words of adjacent sentences), we simply propose to merge the bags-of-words of the mini-batch into a single bag-of-words and perform an update based on that merged bag-of-words. The resulting effect is that each update is as efficient as in stochastic gradient descent, but the number of updates per training epoch is divided by the mini-batch size. As we'll see in the experimental section, this trick produces good word representations, while sufficiently reducing training time. We note that, additionally, we could have used the stochastic approach proposed by Dauphin et al. (2011) for reconstructing binary bag-of-words representations of documents, to further improve the efficiency of training. They use importance sampling to avoid reconstructing the whole V-dimensional input vector.

7.1 Document representations

Once we learn the language specific word representation matrices $\mathbf{W}$ and $\mathbf{V}$ as described above, we can use them to construct document representations, by using their columns as word vector representations. Given a document $\mathbf{d}$, we represent it as the tf-idf weighted sum of its words' representations: $\psi_X(\mathbf{d}) = \mathbf{W}\textit{tf-idf}(\mathbf{d})$ for language X and $\psi_Y(\mathbf{d}) = \mathbf{V}\textit{tf-idf}(\mathbf{d})$ for language Y, where $\textit{tf-idf}(\mathbf{d})$ is the tf-idf weight vector of document $\mathbf{d}$.

We use the document representations thus obtained to train our document classifiers, in the cross-lingual document classification task described in Section 7.3.

7.2 Related Work on Multilingual Word Representations

Recent work that has considered the problem of learning bilingual representations of words usually has relied on word-level alignments. Klementiev et al. (2012) propose to train simultaneously two neural network languages models, along with a regular-

ization term that encourages pairs of frequently aligned words to have similar word embeddings. Thus, the use of this regularization term requires to first obtain word-level alignments from parallel corpora. Zou et al. (2013) use a similar approach, with a different form for the regularizer and neural network language models as in (Collobert et al., 2011). In our work, we specifically investigate whether a method that does not rely on word-level alignments can learn comparably useful multilingual embeddings in the context of document classification.

Looking more generally at neural networks that learn multilingual representations of words or phrases, we mention the work of Gao et al. (2014) which showed that a useful linear mapping between *separately trained* monolingual skip-gram language models could be learned. They too however rely on the specification of pairs of words in the two languages to align. Mikolov et al. (2013) also propose a method for training a neural network to learn useful representations of phrases, in the context of a phrase-based translation model. In this case, phrase-level alignments (usually extracted from word-level alignments) are required. Recently, Hermann and Blunsom (2014b,a), proposed neural network architectures and a margin-based training objective that, as in this work, does not rely on word alignments. We will briefly discuss this work in the experiments section. A tree based bilingual autoencoder with similar objective function is also proposed in (Chandar et al., 2014).

7.3 Experiments

The technique proposed in this work enable us to learn bilingual embeddings which capture cross-language similarity between words. We propose to evaluate the quality of these embeddings by using them for the task of cross-language document classification. We followed closely the setup used by Klementiev et al. (2012) and compare with their method, for which word representations are publicly available[2]. The set up is as follows. A labeled data set of documents in some language X is available to train a classifier, however we are interested in classifying documents in a different language Y at test time. To achieve this, we leverage some bilingual corpora, which is not labeled

[2] `http://klementiev.org/data/distrib/`

with any document-level categories. This bilingual corpora is used to learn document representations that are coherent between languages X and Y. The hope is thus that we can successfully apply the classifier trained on document representations for language X directly to the document representations for language Y. Following this setup, we performed experiments on 3 data sets of language pairs: English/German (EN/DE), English/French (EN/FR) and English/Spanish (EN/ES).

For learning the bilingual embeddings, we used sections of the Europarl corpus (Koehn, 2005) which contains roughly 2 million parallel sentences. We considered 3 language pairs. We used the same pre-processing as used by Klementiev et al. (2012). We tokenized the sentences using NLTK (Bird Steven and Klein, 2009), removed punctuations and lowercased all words. We did not remove stopwords.

As for the labeled document classification data sets, they were extracted from sections of the Reuters RCV1/RCV2 corpora, again for the 3 pairs considered in our experiments. Following Klementiev et al. (2012), we consider only documents which were assigned exactly one of the 4 top level categories in the topic hierarchy (CCAT, ECAT, GCAT and MCAT). These documents are also pre-processed using a similar procedure as that used for the Europarl corpus. We used the same vocabularies as those used by Klementiev et al. (2012) (varying in size between $35,000$ and $50,000$).

Models were trained for up to 20 epochs using the same data as described earlier. We used mini-batch (of size 20) stochastic gradient descent. All results are for word embeddings of size $D = 40$, as in Klementiev et al. (2012). Further, to speed up the training for CorrNet we merged each 5 adjacent sentence pairs into a single training instance, as described earlier. For all language pairs, λ was set to 4. The other hyperparameters were tuned to each task using a training/validation set split of 80% and 20% and using the performance on the validation set of an averaged perceptron trained on the smaller training set portion (notice that this corresponds to a monolingual classification experiment, since the general assumption is that no labeled data is available in the test set language).

We compare our models with the following approaches:

- Klementiev et al. (2012): This model uses word embeddings learned by a mul-

titask neural network language model with a regularization term that encourages pairs of frequently aligned words to have similar word embeddings. From these embeddings, document representations are computed as described in Section 7.1.

- MT: Here, test documents are translated to the language of the training documents using a standard phrase-based MT system, MOSES[3] which was trained using default parameters and a 5-gram language model on the Europarl corpus (same as the one used for inducing our bilingual embeddings).

- Majority Class: Test documents are simply assigned the most frequent class in the training set.

For the EN/DE language pairs, we directly report the results from Klementiev et al. (2012). For the other pairs (not reported in Klementiev et al. (2012)), we used the embeddings available online and performed the classification experiment ourselves. Similarly, we generated the MT baseline ourselves.

Table 7 summarizes the results. They were obtained using 1000 RCV training examples. We report results in both directions, *i.e.* language X to Y and vice versa. The best performing method in all the pairs except one is CorrNet. In particular, CorrNet often outperforms the approach of Klementiev et al. (2012) by a large margin.

In the last row of the table, we also include the results of some recent work by Hermann and Blunsom (2014b,a). They proposed two neural network architectures for learning word and document representations using sentence-aligned data only. Instead of an autoencoder paradigm, they propose a margin-based objective that aims to make the representation of aligned sentences closer than non-aligned sentences. While their trained embeddings are not publicly available, they report results for the EN/DE classification experiments, with representations of the same size as here ($D = 40$) and trained on 500K EN/DE sentence pairs. Their best model in that setting reaches accuracies of 83.7% and 71.4% respectively for the EN $\rightarrow$ DE and DE $\rightarrow$ EN tasks. One clear advantage of our model is that unlike their model, it can use additional monolingual data. Indeed, when we train CorrNet with 500k EN/DE sentence pairs, plus

[3] `http://www.statmt.org/moses/`

monolingual RCV documents (which come at no additional cost), we get accuracies of 87.9% (EN $\rightarrow$ DE) and 76.7% (DE $\rightarrow$ EN), still improving on their best model. If we do not use the monolingual data, CorrNet's performance is worse but still competitive at 86.1% for EN $\rightarrow$ DE and 68.8% for DE $\rightarrow$ EN. Finally, without constraining D to 40 (they use 128) and by using additional French data, the best results of Hermann and Blunsom (2014b) are 88.1% (EN $\rightarrow$ DE) and 79.1% (DE $\rightarrow$ EN), the later being, to our knowledge, the current state-of-the-art (as reported in the last row of Table 7).

	EN $\rightarrow$ DE	DE $\rightarrow$ EN	EN $\rightarrow$ FR	FR $\rightarrow$ EN	EN $\rightarrow$ ES	ES $\rightarrow$ EN
CorrNet	**91.8**	74.2	**84.6**	**74.2**	49.0	**64.4**
Klementiev et al.	77.6	71.1	74.5	61.9	31.3	63.0
MT	68.1	67.4	76.3	71.1	**52.0**	58.4
Majority Class	46.8	46.8	22.5	25.0	15.3	22.2
Hermann and Blunsom	88.1	**79.1**	N.A.	N.A.	N.A.	N.A.

Table 7: Cross-lingual classification accuracy for 3 language pairs, with 1000 labeled examples.

We also evaluate the effect of varying the amount of supervised training data for training the classifier. For brevity, we report only the results for the EN/DE pair, which are summarized in Figure 5 and Figure 6. We observe that CorrNet clearly outperforms the other models at almost all data sizes. More importantly, it performs remarkably well at very low data sizes (100), suggesting it learns very meaningful embeddings, though the method can still benefit from more labeled data (as in the DE $\rightarrow$ EN case).

Table 8 also illustrates the properties captured within and across languages, for the EN/DE pair. For a few English words, the words with closest word representations (in Euclidean distance) are shown, for both English and German. We observe that words that form a translation pair are close, but also that close words within a language are syntactically/semantically similar as well.

january		president		said	
en	de	en	de	en	de
january	januar	president	präsident	said	gesagt
march	märz	i	präsidentin	told	sagte
october	oktober	mr	präsidenten	say	sehr
july	juli	presidents	herr	believe	heute
december	dezember	thank	ich	saying	sagen
1999	jahres	president-in-office	ratspräsident	wish	heutigen
june	juni	report	danken	shall	letzte
month	1999	voted	danke	again	hier
oil		microsoft		market	
en	de	en	de	en	de
oil	öl	microsoft	microsoft	market	markt
supply	boden	cds	cds	markets	marktes
supplies	befindet	insider	warner	single	märkte
gas	gerät	ibm	tageszeitungen	commercial	binnenmarkt
fuel	erdöl	acquisitions	ibm	competition	märkten
mineral	infolge	shareholding	handelskammer	competitive	handel
petroleum	abhängig	warner	exchange	business	öffnung
crude	folge	online	veranstalter	goods	binnenmarktes

Table 8: Example English words along with 8 closest words both in English (en) and German (de), using the Euclidean distance between the embeddings learned by CorrNet

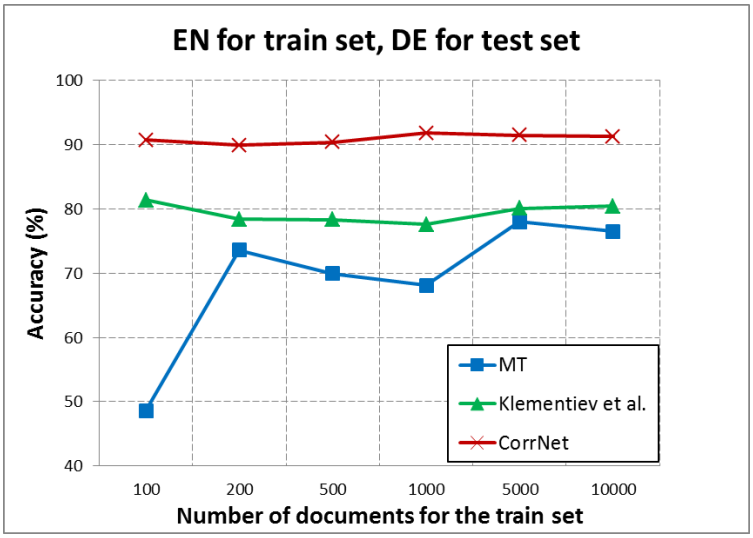

Figure 5: Cross-lingual classification accuracy results for EN → DE

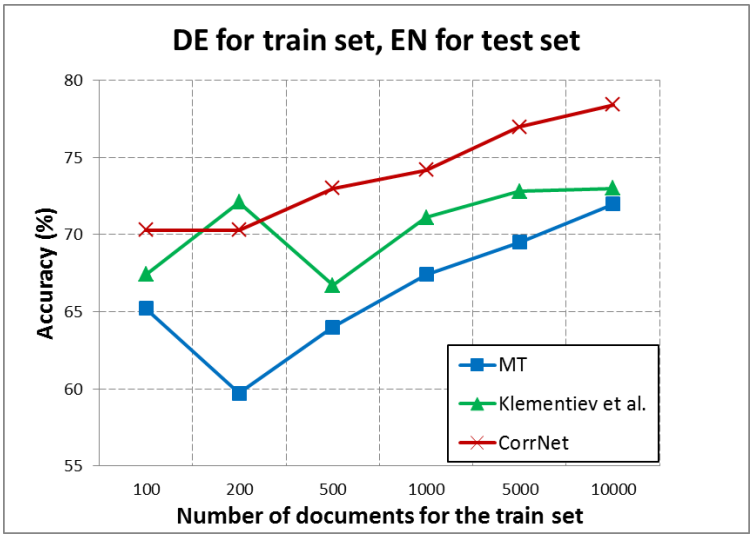

Figure 6: Cross-lingual classification accuracy results for DE → EN

The excellent performance of CorrNet suggests that merging several sentences into single bags-of-words can still yield good word embeddings. In other words, not only we do not need to rely on word-level alignments, but exact sentence-level alignment is also not essential to reach good performances. We experimented with the merging of 5, 25 and 50 adjacent sentences into a single bag-of-words. Results are shown in Table 9. They suggest that merging several sentences into single bags-of-words does

not necessarily impact the quality of the word embeddings. Thus they confirm that exact sentence-level alignment is not essential to reach good performances as well.

	# sent.	EN → DE	DE → EN	EN → FR	FR → EN	EN → ES	ES → EN
CorrNet	5	91.75	72.78	84.64	74.2	49.02	64.4
	25	88.0	64.5	78.1	70.02	68.3	54.68
	50	90.2	49.2	82.44	75.5	38.2	67.38

Table 9: Cross-lingual classification accuracy for 3 different pairs of languages, when merging the bag-of-words for different numbers of sentences. These results are based on 1000 labeled examples.

8 Transliteration Equivalence

In the previous section, we showed the application of CorrNet in a cross language learning setup. In addition to cross language learning, CorrNet can also be used for matching equivalent items across views. As a case study, we consider the task of determining transliteration equivalence of named entities wherein given a word u written using the script of language X and a word v written using the script of language Y the goal is to determine whether u and v are transliterations of each other. Several approaches have been proposed for this task and the one most related to our work is an approach which uses CCA for determining transliteration equivalence.

We condider English-Hindi as the language pair for which transliteration equivalence needs to be determined. For learning common representations we used approximately 15,000 transliteration pairs from NEWS 2009 English-Hindi training set (Li et al., 2009). We represent each Hindi word as a bag of 2860 bigram characters. This forms the first view ($\mathbf{x}_i$). Similarly we represent each English word as a bag of 651 bigram characters. This forms the second view ($\mathbf{y}_i$). Each such pair $(\mathbf{x}_i, \mathbf{y}_i)$ then serves as one training instance for the CorrNet.

For testing we consider the standard NEWS 2010 transliteration mining test set (Kumaran et al., 2010). This test set contains approximately 1000 Wikipedia English

Hindi title pairs. The original task definition is as follows. For a given English title containing T_1 words and the corresponding Hindi title containing T_2 words identify all pairs which form a transliteration pair. Specifically, for each title pair, consider all $T_1 \times T_2$ word pairs and identify the correct transliteration pairs. In all, the test set contains 5468 word pairs out of which 982 are transliteration pairs. For every word pair $(\mathbf{x}_i, \mathbf{y}_i)$ we obtain a 50 dimensional common representation for $\mathbf{x}_i$ and $\mathbf{y}_i$ using the trained CorrNet. We then calculate the correlation between the representations of $\mathbf{x}_i$ and $\mathbf{y}_i$. If the correlation is above a threshold we mark the word pair as equivalent. This threshold is tuned using an additional 1000 pairs which were provided as training data for the NEWS 2010 transliteration mining task. As seen in Table 10 CorrNet clearly performs better than the other methods. Note that our aim is not to achieve state of the art performance on this task but to compare the quality of the shared representations learned using different CRL methods considered in this paper.

Model	F1-measure (%)
CCA	49.68
KCCA	42.36
MAE	72.75
CorrNet	**81.56**

Table 10: Performance on NEWS 2010 En-Hi Transliteration Mining Dataset

9 Conclusion and Future Work

In this paper, we proposed Correlational Neural Networks as a method for learning common representations for two views of the data. The proposed model has the capability to reconstruct one view from the other and it ensures that the common representations learned for the two views are aligned and correlated. Its training procedure is also scalable. Further, the model can benefit from additional single view data, which is often available in many real world applications. We employ the common representations learned using CorrNet for two downstream applications, *viz.*, cross language document

classification and transliteration equivalence detection. For both these tasks we show that the representations learned using CorrNet perform better than other methods.

We believe it should be possible to extend CorrNet to multiple views. This could be very useful in applications where varying amounts of data are available in different views. For example, typically it would be easy to find parallel data for English/German and English/Hindi, but harder to find parallel data for German/Hindi. If data from all these languages can be projected to a common subspace then English could act as a pivot language to facilitate cross language learning between Hindi and German. We intend to investigate this direction in future work.

Acknowledgement

We would like to thank Alexander Klementiev and Ivan Titov for providing the code for the classifier and data indices for the cross language document classification task. We would like to thank Janarthanan R for valuable discussions and inputs.

References

S. Akaho. A kernel method for canonical correlation analysis. *In Proc. Int'l Meeting on Psychometric Society*, 2001.

Galen Andrew, Raman Arora, Jeff Bilmes, and Karen Livescu. Deep canonical correlation analysis. *ICML*, 2013.

Raman Arora and Karen Livescu. Kernel CCA for multi-view learning of acoustic features using articulatory measurements. In *2012 Symposium on Machine Learning in Speech and Language Processing, MLSLP 2012, Portland, Oregon, USA, September 14, 2012*, pages 34–37, 2012.

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010.

Edward Loper Bird Steven and Ewan Klein. *Natural Language Processing with Python*. OReilly Media Inc., 2009.

Sarath Chandar, Mitesh M. Khapra, Balaraman Ravindran, Vikas C. Raykar, and Amrita Saha. Multilingual deep learning. *NIPS Deep Learning Workshop*, 2013.

Sarath Chandar, Stanislas Lauly, Hugo Larochelle, Mitesh M Khapra, Balaraman Ravindran, Vikas Raykar, and Amrita Saha. An autoencoder approach to learning bilingual word representations. In *Proceedings of NIPS*, 2014.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research*, 12:2493–2537, 2011.

Yann Dauphin, Xavier Glorot, and Yoshua Bengio. Large-Scale Learning of Embeddings with Reconstruction Sampling. In *Proceedings of the 28th International Conference on Machine Learning (ICML 2011)*, pages 945–952. Omnipress, 2011.

Paramveer Dhillon, Dean Foster, and Lyle. Ungar. Multi-view learning of word embeddings via cca. *In NIPS*, 2011.

Jianfeng Gao, Xiaodong He, Wen-tau Yih, and Li Deng. Learning continuous phrase representations for translation modeling. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 699–709, Baltimore, Maryland, June 2014.

Karl Moritz Hermann and Phil Blunsom. Multilingual Distributed Representations without Word Alignment. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2014a.

Karl Moritz Hermann and Phil Blunsom. Multilingual models for compositional distributed semantics. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 58–68, 2014b.

H. Hotelling. Relations between two sets of variates. *Biometrika*, 28:321 – 377, 1936.

W.W. Hsieh. Nonlinear canonical correlation analysis by neural networks. *Neural Networks*, 13(10):1095 – 1105, 2000.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015. URL `http://arxiv.org/abs/1502.03167`.

Davis E. King. Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10:1755–1758, 2009.

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. Inducing Crosslingual Distributed Representations of Words. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, 2012.

Philipp Koehn. Europarl: A parallel corpus for statistical machine translation. In *MT Summit*, 2005.

A Kumaran, Mitesh M. Khapra, and Haizhou Li. Report of news 2010 transliteration mining shared task. In *Proceedings of the 2010 Named Entities Workshop*, pages 21–28, Uppsala, Sweden, July 2010.

Haizhou Li, A Kumaran, Min Zhang, and Vladimir Pervouvhine. Whitepaper of news 2009 machine transliteration shared task. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 19–26, Suntec, Singapore, August 2009.

Yichao Lu and Dean P. Foster. large scale canonical correlation analysis with iterative least squares. *In NIPS*, 2014.

Tomas Mikolov, Quoc Le, and Ilya Sutskever. Exploiting Similarities among Languages for Machine Translation. Technical report, arXiv, 2013.

J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and Ng. Andrew. Multimodal deep learning. *ICML*, 2011.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau,

M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Raghavendra Udupa and Mitesh M. Khapra. Transliteration equivalence using canonical correlation analysis. In *Proceedings of the 32nd European Conference on IR Research*, pages 75–86, 2010.

Will Y. Zou, Richard Socher, Daniel Cer, and Christopher D. Manning. Bilingual Word Embeddings for Phrase-Based Machine Translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*, 2013.