Correlational Neural Networks

Sarath Chandar
apsarathchandar@gmail.com
University of Montreal, Montreal QC H3T 1J4, Canada

Mitesh M. Khapra
mikhapra@in.ibm.com
IBM Research India, Bangalore 560077, India

Hugo Larochelle
hugo.larochelle@usherbrooke.ca
University of Sherbrooke, Sherbrooke QC J1K 2R1, Canada

Balaraman Ravindran
ravi@cse.iitm.ac.in
Indian Institute of Technology Madras, Chennai, India, 600036

Common representation learning (CRL), wherein different descriptions (or views) of the data are embedded in a common subspace, has been 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, 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 AE and CCA with respect to its ability to learn correlated common representations. We employ CorrNet for several cross-language tasks and show that the representations learned using it perform better than the ones learned using other state-of-the-art approaches.

This work was done while Sarath Chandar was at IIT Madras and IBM Research India.

doi:10.1162/NECO_a_00801

1 Introduction

In several real-world application, the data contain 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, but not subtitles, may be available. Recently there has been, interest in learning a common representation for multiple views of the data (Ngiam et al., 2011; Klementiev, Titov, & Bhattarai, 2012; Chandar, Lauly, Larochelle, Khapra, Ravindran, Raykar, & Saha, 2013; Chandar et al., 2014; Andrew, Arora, Bilmes, & Livescu, 2013; Hermann & Blunsom, 2014b; Wang, Arora, Livescu, & Bilmes, 2015), which can be useful in several downstream applications when some of the views are missing. We consider four applications to motivate the importance of learning common representations: (1) reconstruction of a missing view, (2) transfer learning, (3) matching corresponding items across views, and (4) improving single-view performance by using data from other 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 and video are 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 or 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. Third, 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. Finally, consider the case where we are interested in learning word representations for a language. If we have access to translations of these words in another language, then these translations can provide some context for disambiguation, which can lead to learning better word representations. In other words, jointly learning representations for a word in language L_1 and its translation in language L_2 can lead to better word representations in L_1 (see section 9).

Having motivated the importance of common representation learning (CRL), we now formally define this task. Consider some data $\mathcal{Z} = \{\mathbf{z}_i\}_{i=1}^{N}$

with 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 & Khapra, 2010; Dhillon, Foster, & Ungar, 2011). By definition, it aims to produce correlated common representations, but it suffers from some drawbacks. First, it is not easily scalable to very large data sets. Of course, some approaches try to make CCA scalable (Lu & Foster, 2014), but such scalability comes at the cost of performance. Further, since CCA does not explicitly focus on reconstruction, reconstructing one view from the other might result in low-quality reconstruction. Finally, CCA cannot benefit from additional nonparallel, 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 are 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 or 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), which makes the representations learned to be predictive of each other. However, 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 unsuitable for transfer learning, since the views are not guaranteed to be projected to a common subspace. This is indeed verified by the results reported by Ngiam et al. (2011), who find 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 article, we propose correlational neural network (CorrNet) as a method for learning common representations that 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 and 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 minibatches. 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 it useful in many real-world applications where additional single-view data are available.

We evaluate CorrNet using four experimental setups. First, we use the MNIST handwritten digit recognition data set to compare CorrNet with other state-of-the-art CRL approaches. In particular, we evaluate its (1) ability to self-/or cross-reconstruct, (2) ability to produce correlated common representations, and (3) 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 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. Third, 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. Finally, we employ CorrNet for a bigram similarity task and show that jointly learning word representations for two languages (two views) leads to better word representations. Specifically, representations learned using CorrNet help to improve the performance of a bigram similarity task. We emphasize that unlike models that have been tested mostly in only one of these scenarios, we demonstrate the effectiveness of CorrNet in all these different scenarios.

The remainder of this article 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, 8, and 9, we report results obtained by using CorrNet for the tasks of cross-language document classification, transliteration equivalence detection, and bigram similarity respectively. Finally, we present concluding remarks in section 10 and highlight possible future work.

2 Correlational Neural Network

Our aim is to learn a common representation from two views of the same data such that (1) any single view can be reconstructed from the common representation, (2) a single view can be predicted from the representation of another view, and (3) like CCA, the representations learned for the two views are correlated. The first goal 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 that can work with two views of the data while being explicitly trained to achieve all the above goals. In the following sections, we describe our model and the training procedure.

2.1 Model. We start by proposing a neural network architecture that 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

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

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 nonlinear 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

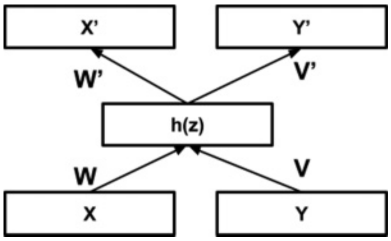

Figure 1: Correlational neural network.

illustrated in Figure 1. The parameters of the model are $\theta = \{\mathbf{W}, \mathbf{V}, \mathbf{W}', \mathbf{V}', \mathbf{b}, \mathbf{b}'\}$. In section 2.2, 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, that is, minimize the error in reconstructing $\mathbf{x}_i$ from $\mathbf{x}_i$ and $\mathbf{y}_i$ from $\mathbf{y}_i$.
- Minimize the cross-reconstruction error, that is, 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:

$$\begin{aligned}\mathcal{J}_{\mathcal{Z}}(\theta) &= \sum_{i=1}^{N} (L(\mathbf{z}_i, g(h(\mathbf{z}_i))) + L(\mathbf{z}_i, g(h(\mathbf{x}_i))) \\ &\quad + L(\mathbf{z}_i, g(h(\mathbf{y}_i)))) - \lambda \operatorname{corr}(h(X), h(Y)) \\ \operatorname{corr}(h(X), h(Y)) &= \frac{\sum_{i=1}^{N}(h(\mathbf{x}_i) - \overline{h(X)})(h(\mathbf{y}_i) - \overline{h(Y)})}{\sqrt{\sum_{i=1}^{N}(h(\mathbf{x}_i) - \overline{h(X)})^2 \sum_{i=1}^{N}(h(\mathbf{y}_i) - \overline{h(Y)})^2}},\end{aligned}$$

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, we use cross-entropy as the reconstruction error; otherwise, we use squared error loss as the reconstruction error. For simplicity, we use the shorthand $h(\mathbf{x}_i)$ and $h(\mathbf{y}_i)$ to note the representations

$h((\mathbf{x}_i, 0))$ and $h((0, \mathbf{y}_i))$ that are based on only a single view.[1] For each data point with two views x and y, $h(\mathbf{x}_i)$ just means that we are computing the hidden representation using only the x view. In other words, in equation $\mathrm{h}(\mathbf{z}) = \mathrm{f}(\mathbf{W}\mathbf{x} + \mathbf{V}\mathbf{y} + \mathbf{b})$, we set $\mathbf{y} = 0$. So, $\mathrm{h}(\mathbf{x}) = \mathrm{f}(\mathbf{W}\mathbf{x} + \mathbf{b})$. $h(\mathbf{x}) = h(\mathbf{x}, 0)$ is not a choice per se, but a notation we are defining. $h(\mathbf{z})$, $h(\mathbf{x})$, and $h(\mathbf{y})$ are certainly not guaranteed to be identical, though training will gain in making them that way because of the various reconstruction terms. The correlation term in the objective function is calculated considering the hidden representation as a random vector.

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 minibatch 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 four hyperparameters: (1) the number of units in its hidden layer, (2) λ, (3) minibatch size, and (4) 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 third hyperparameter approximates the correlation of the entire data set; larger minibatches are preferred over smaller minibatches. 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 few two-view

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

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 these data in order to improve the learned representation. CorrNet can achieve this by using the single-view data to improve the self-reconstruction error, as we explain.

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 ($\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{Z}$), we construct three sets of minibatches by sampling data from $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{Z}$, respectively. We then feed these minibatches 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 nonlinear 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 **W**, **V**, and **b**.
2. Modify the CorrNet model such that the first input view connects to a hidden layer using weights **W** and bias **b**. Similarly, connect the second view to a hidden layer using weights **V** and bias **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.

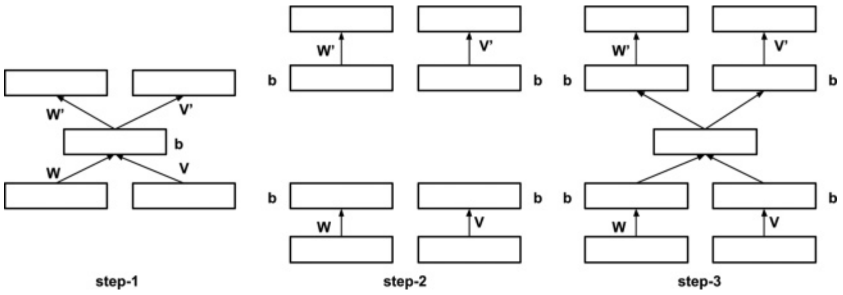

Figure 2: Stacking CorrNet to create a deep correlational neural network.

We point out that we could have followed the procedure described in Chandar et al. (2013) for training Deep CorrNet. Chandar et al. (2013) learn deep representation for each view separately and use it along with a shallow CorrNet to learn a common representation. However, feeding nonlinear 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 they used during pretraining and training is different. Specifically, during pretraining, 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, the objectives during training and pretraining are aligned.

Our current training procedure for deep CorrNet is similar to greedy layerwise pretraining of deep autoencoders. We believe that this procedure is more faithful to the global training objective of CorrNet, and it works well. We do not have strong empirical evidence that this is superior to other methods such as the one described in Chandar et al. (2013) and Ngiam et al. (2011). When we have fewer parallel data, using the method described in Chandar et al. (2013) makes more sense, though each method has its advantages. We leave a detailed comparison of these different alternatives of Deep CorrNet as future work.

4 Related Models

In this section, we describe other related models for common representation learning. We restrict our discussion to CCA-based and neural network–based methods.

Canonical correlation analysis (CCA) (Hotelling, 1936) and its variants, such as regularized CCA (Vinod, 1976; Nielsen, Hansen, & Strother, 1998; Cruz-Cano & Lee, 2014), are the de facto approaches used for learning common representation for two different views in the literature (Udupa & Khapra, 2010; Dhillon et al., 2011). Kernel CCA (Akaho, 2001; Hardoon, Szedmak, & Shawe-Taylor, 2004), another variant of CCA, uses the standard kernel trick to find pairs of nonlinear projections of the two views. Deep CCA, a deep version of CCA is also introduced in Andrew et al. (2013). One issue with CCA is that it is not easily scalable. Although there are several works on scaling CCA (see Lu & Foster, 2014), they are all approximations to CCA and hence lead to a decrease in performance. Also it is not trivial to extend CCA to multiple views. However, there is some recent work along this line (Tenenhaus & Tenenhaus, 2011; Luo, Tao, Wen, Ramamohanarao, & Xu, 2015) which require complex computations. Finally, conventional CCA-based models can work only with parallel data. However, in real-life situations, parallel data are costly when compared to single-view data. The inability of CCA to leverage such single-view data acts as a drawback in many real-world applications. Representation-constrained CCA (RCCCA) (Mishra, 2009) is one such model that can benefit from both single-view and multiview data. It effectively uses a weighted combination of PCA (for single view) and CCA (for two views) by minimizing self-reconstruction errors and maximizing correlation. CorrNet, in contrast, minimizes both self- and cross-reconstruction error while maximizing correlation. RCCCA can also be considered a linear version of DCCAE proposed in Wang et al. (2015).

Hsieh (2000) is one of the earliest neural network–based models 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 learned 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 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. Although the architecture of MAE is similar to that of CorrNet, there are clear differences in the training procedure they use. First, MAE aims only to minimize the following three errors: (1) error in reconstructing z_i from x_i (E_1), (2) error in reconstructing z_i from y_i (E_2), and (3) error in reconstructing z_i from z_i (E_3). More specifically, unlike the fourth term in our objective function, the objective function MAE uses does not contain any term that forces the network to learn correlated common representations. Second, there is a difference in the manner in which these terms are considered during training. Unlike CorrNet, MAE considers only 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 that considers all three loss terms together performs better than the one that considers them separately (see refer section 5.5).

Deep canonical correlation analysis (DCCA) (Andrew et al., 2013) is a recently proposed neural network approach for CCA. It 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. Deep canonically correlated auto encoders (DCCAE) (Wang et al., 2015) (developed in parallel with our work) is an extension of DCCA that considers self-reconstruction and correlation. Unlike CorrNet, it does not consider cross-reconstruction.

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 the

- 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 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 data set for all our experiments. This data consist of 60,000 training images and 10,000 test images. Each image is a 28×28 matrix of pixels, each pixel representing one of 256 gray-scale values. We treated the left half of the image as one view and the right half as another image. Thus, each view contains $14 \times 28 = 392$ dimensions. We split the training 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 hyperparameters of the four models described.

5.2 Performance of Self- and Cross-Reconstruction. Among the four models listed above, only CorrNets and MAE have been explicitly trained to construct a view from itself as well as from the other view. So in this 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.

Table 1: Mean Squared Error for CorrNet and MAE for Self-Reconstruction and Cross-Reconstruction.

Model	MSE for Self-Reconstruction	MSE for Cross-Reconstruction
CorrNet	3.6	4.3
MAE	**2.1**	**4.2**

Note: The best results are in bold.

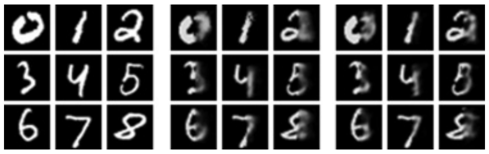

Figure 3: Reconstruction of right half of the image given the left half. The left block shows the original images, the middle block shows images where the right half is reconstructed by CorrNet, and the right block shows images where the right half is reconstructed by MAE.

Table 1 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, the emphasis in CorrNet 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 section 5.3, the embeddings learned 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.

5.3 Correlation between Representations of Two Views. In CorrNet we emphasize 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 learned 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.

The total correlation captured in the 50 dimensions learned 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).

Table 2: Sum/Total Correlation Captured in the 50 Dimensions of the Common Representations Learned by Different Models Using MNIST Data.

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

Note: The best results are in bold.

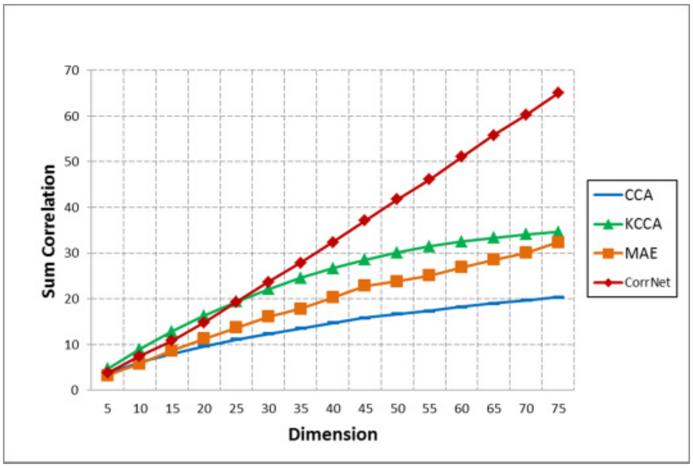

Figure 4: Sum/total correlation as a function of the number of dimensions in the common representations learned by different models using MNIST data.

For all the models, we tuned the hyperparameters for $dim = 50$ and used the same hyperparameters for all dimensions. 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 on the task of predicting digits from only 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 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

Table 3: Transfer Learning Accuracy Using the Representations Learned Using Different Models on the MNIST Data Set.

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**
Single view	81.62	80.06

Note: The best results are in bold.

Table 4: Transfer Learning Accuracy Using the Representations Learned Using Different Models Trained with 10,000 Instances from the MNIST Data Set.

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**
Single view	81.62	80.06

Note: The best results are in bold.

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. For all the models considered in this experiment, representation learning is done using 50,000 training images, and the best hyperparameters are chosen using the 10,000 images from the validation set. With the chosen model, we report five-fold cross-validation accuracy using 10,000 images available in the standard test set of MNIST data. We report accuracy for two settings left to right (training on left view, testing on right view) and right to left (training on right view, testing on left view) (see Table 3).

Single view model corresponds to the classifier trained and tested on same view. This is the upper bound for the performance of any transfer learning algorithm. Once again, we see that CorrNet performs significantly better than the other models. We verify that this holds even when we decrease the data for learning common representation to 10,000 images. The results, reported in Table 4, show that even with fewer data, CorrNet performs better than other models.

5.5 Relation with MAE. On the face of it, it may seem that both CorrNet and MAE differ only in their objective functions. Specifically, if we remove the last correlation term from the objective function of CorrNet, then it would become equivalent to MAE. To verify this, we conducted

Table 5: Results for Transfer Learning across Views.

Model	Optimization	Left to Right	Right to Left
MAE	SGD	63.9	67.98
CorrNet(123)	SGD	63.89	67.93
MAE	RMSProp	64.14	68.88
CorrNet(123)	RMSProp	67.82	72.13

experiments using both MAE and CorrNet without the last term (say CorrNet(123)). When using SGD to train the networks, we found that the performance is almost similar. However, when we use some advanced optimization technique like RMSProp, CorrNet(123) starts performing better than MAE. The results are reported in Table 5.

This experiment sheds some light on why CorrNet is better than MAE. Although the objective of MAE and CorrNet(123) is the same, MAE tries to solve it in a stochastic way, which adds more noise. However, CorrNet(123) performs better since it is actually working on the combined objective function and not the stochastic version (one term at a time) of it.

5.6 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 that contain different combinations of these terms. In addition, we consider four more loss terms for our analysis:

- $L_5 = \sum_{i=1}^{N} L(\mathbf{y}_i, g(h(\mathbf{x}_i))$
- $L_6 = \sum_{i=1}^{N} L(\mathbf{x}_i, g(h(\mathbf{y}_i))$
- $L_7 = \sum_{i=1}^{N} L(\mathbf{x}_i, g(h(\mathbf{x}_i))$
- $L_8 = \sum_{i=1}^{N} L(\mathbf{y}_i, g(h(\mathbf{y}_i))$,

where L_5 and L_6 essentially capture the loss in reconstructing only one view (say, $\mathbf{x}_i$) from the other view ($\mathbf{y}_i$) while L_7 and L_8 capture the loss in self-reconstruction.

For this, we first learn common representations using different loss functions, as listed in the first column of Table 6. We then repeated the transfer learning experiments using common representations learned from each of

Table 6: Comparison of the Performance of Transfer Learning with Representations Learned Using Different Loss Functions.

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
$L_7 + L_8$	10.26	10.33
$L_7 + L_8 + L_4$	73.03	76.08

Note: The best results are in bold.

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.

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 the correlation term (L_4) in the loss function clearly produces representations that 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 10,000 images from the MNIST training data set, 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 five-fold cross-validation using 10,000 images from MNIST test data. We report results for two settings: left to right (training on left view, testing on right view) and right to left (training on right view, testing on left view). These results are summarized in Table 7. 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

Table 7: Comparison of Sum Correlation and Transfer Learning Performance of Different Deep Models.

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

Note: The best results are in bold.

hidden layers containing 500, 300, and 50 units respectively. The third layer, containing 50 units, is used as the common representation.

Both Deep CorrNets (CorrNet-500-50 and CorrNet-500-300-50) clearly perform better than the corresponding DCCA. We notice that for both transfer learning tasks, the three-layered CorrNet (CorrNet-500-300-50), performs better than the two-layered CorrNet (CorrNet-500-50), but the sum correlation of the two-layered CorrNet is better than that of the three-layered CorrNet.

7 Cross-Language Document Classification

In this section, we 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, we 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 ith vocabulary word is present in the sentence and 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 that 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 nonlinearity. Thus, we can view $\mathbf{W}$ as a matrix whose columns act as vector representations (embeddings) for each word.

We 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 would 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 **W** and **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 languages, 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 dimensionality corresponds to the size of the vocabulary, which is typically large), reconstructing each bag 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 minibatch training (where a minibatch contains a list of the bags of words of adjacent sentences), we simply propose to merge the bags of words of the minibatch 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 minibatch size. As we will see in the section 7.3, this trick produces good word representations while sufficiently reducing training time. We note in addition, that we could have used the stochastic approach proposed by Dauphin, Glorot, and Bengio (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 **W** and **V** as described above, we can use them to construct document representations by using their columns as word vector representations. Given a document **d**, we represent it as the tf-idf weighted sum of its words' representations: $\psi_X(\mathbf{d}) = \mathbf{W}tf - idf(\mathbf{d})$ for language X and $\psi_Y(\mathbf{d}) = \mathbf{V}tf - idf(\mathbf{d})$ for language Y, where $tf - idf(\mathbf{d})$ is the tf-idf weight vector of document **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 regularization term that encourages pairs of frequently aligned words to have similar word embeddings. Thus, the use of this regularization term requires first obtaining word-level alignments from parallel corpora. Zou, Socher, Cer, and Manning (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, Le, and Sutskever (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 (2014a, 2014b), proposed neural network architectures and a margin-based training objective that, as in this work, does not rely on word alignments. We briefly discuss this work in section 7.3. 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 enables us to learn bilingual embeddings that 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 ours with their method, for which word representations are publicly available.[2] The setup 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 not labeled 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 three 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 three language pairs. We used the same preprocessing as did Klementiev et al. (2012). We tokenized the sentences using NLTK (Bird Steven & Klein, 2009), removed punctuation, and lowercased all words. We did not remove stopwords.

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

The labeled document classification data sets were extracted from sections of the Reuters RCV1/RCV2 corpora, again for the three pairs considered in our experiments. Following Klementiev et al. (2012), we consider only documents that were assigned exactly one of the four top-level categories in the topic hierarchy (CCAT, ECAT, GCAT, and MCAT). These documents are also preprocessed 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 minibatch (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 five adjacent sentence pairs into a single training instance, as described earlier. For all language pairs, λ was set to four. 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 are 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 multitask 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] trained using default parameters and a five-gram language model on the Europarl corpus (the 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 that work), we used the embeddings available online and performed the classification experiment ourselves. Similarly, we generated the MT baseline ourselves.

Table 8 summarizes the results. They were obtained using 1000 RCV training examples. We report results in both directions: 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.

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

Table 8: Cross-Lingual Classification Accuracy for Three Language Pairs, with 1000 Labeled Examples.

	EN → DE	DE → EN	EN → FR	FR → EN	EN → ES	ES → EN
CorrNet	**91.8**	74.2	**84.6**	**74.2**	49.0	**64.4**
Klementiev et al. (2012)	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 (2014b)	88.1	**79.1**	NA	NA	NA	NA

Note: The best results are in bold.

In the last row of the table, we also include the results of some recent work by Hermann and Blunsom (2014a, 2014b). 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 nonaligned 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 500,000 EN/DE sentence pairs. Their best model in that setting reaches accuracies of 83.7% and 71.4%, respectively, for the EN → DE and DE → 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 500,000 EN/DE sentence pairs, plus monolingual RCV documents (which come at no additional cost), we get accuracies of 87.9% (EN → DE) and 76.7% (DE → 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 → DE and 68.8% for DE → 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 → DE) and 79.1% (DE → EN), the latter being, to our knowledge, the current state-of-the-art (as reported in the last row of Table 8).[4]

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 Figures 5 and 6. We observe that CorrNet clearly outperforms the other models at almost all data sizes. More important, it performs remarkably well at very low data sizes (100), suggesting it learns meaningful embeddings, though the method can still benefit from more labeled data (as in the DE → EN case).

[4]Since we published our results in Chandar et al. (2014), Soyer, Stenetorp, and Aizawa (2015) have improved the performance for EN→DE and DE→EN to 92.7% and 82.4%, respectively.

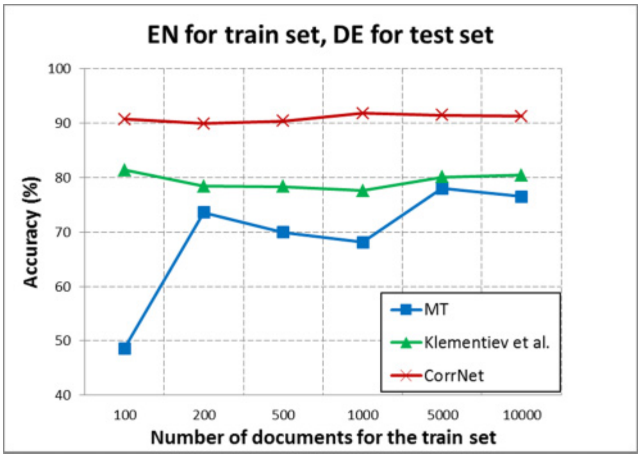

Figure 5: Cross-lingual classification accuracy results for EN → DE.

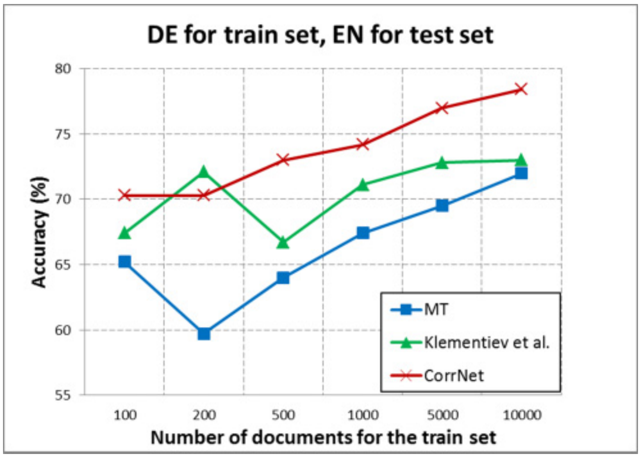

Figure 6: Cross-lingual classification accuracy results for DE → EN.

Table 9 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 or semantically similar as well.

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 do we not need to rely on word-level alignments,

Table 9: Example English Words with Eight Closest Words in English and German, Using the Euclidean Distance between the Embeddings Learned by CorrNet.

January		President		Said	
English	German	English	German	English	German
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	
English	German	English	German	English	German
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

but exact sentence-level alignment is also not essential to reach good performance. We experimented with the merging of 5, 25, and 50 adjacent sentences into a single bag of words. Results are shown in Table 10. They suggest that merging several sentences into single bags of words does not necessarily affect the quality of the word embeddings. Thus, they confirm that exact sentence-level alignment is not essential to reach good performances as well.

8 Transliteration Equivalence

In the previous section, we showed the application of CorrNet in a cross-language learning setup. In addition, CorrNet can also be used for matching equivalent items across views. As a case study, we consider the task of determining the 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

Table 10: Cross-Lingual Classification Accuracy for Three Different Pairs of Languages When Merging the Bag of Words for Different Numbers of Sentences.

	Number of Sentences	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

Note: These results are based on 1000 labeled examples.

Table 11: Performance on NEWS 2010 En-Hi Transliteration Mining Data Set.

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

Note: The best result is in bold.

this task. The one most related to our work uses CCA for determining transliteration equivalence.

We consider 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 the NEWS 2009 English-Hindi training set (Li, Kumaran, Zhang, & Pervouvhine, 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, Khapra, & Li, 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 that 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 provided as training data for the NEWS 2010 transliteration mining task. As seen in Table 11, 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 article.

9 Bigram Similarity Using Multilingual Word Embedding

In this section, we consider one more data set/application to compare the performance of CorrNet with other state-of-the-art methods. Specifically, the task at hand is to calculate the similarity score between two bigram pairs in English based on their representations. These representations are calculated from word representations learned using English-German word pairs. The motivation here is that the German word provides some context for disambiguating the English word and hence leads to better word representations. This task has been already considered in Mitchell and Lapata (2010), Lu, Wang, Bansal, Gimpel, and Livescu (2015), and Wang et al. (2015). We follow the similar setup as Wang et al. (2015) and use the same data set. The English and German words are first represented using 640-dimensional monolingual word vectors trained via latent semantic indexing (LSI) on the WMT 2011 monolingual news corpora. We used 36,000 such English-German monolingual word vector pairs for common representation learning. Each pair consisting of one English (x_i) and one German(y_i) word thus acts as one training instance, $z_i = (x_i, y_i)$, for the CorrNet. Once a common representation is learned, we project all the English words into this common subspace and use these word embeddings for computing the similarity of bigram pairs in English.

The bigram similarity data set was initially used in Mitchell and Lapata (2010). We consider the adjective-noun (AN) and verb-object (VN) subsets of the bigram similarity dataset. We use the same tuning and test splits of size 649/1972 for each subset. The vector representation of a bigram is computed by simply adding the vector representations of the two words in the bigram. Following previous work, we compute the cosine similarity between the two vectors of each bigram pair, order the pairs by similarity, and report the Spearman's correlation (ρ) between the model's ranking and human rankings.

Following Wang et al. (2015), we fix the dimensionality of the vectors at $L = 384$. Other hyperparameters are tuned using the tuning data. The results are reported in Table 12, where we compare CorrNet with different methods proposed in Wang et al. (2015). CorrNet performs better than the previous state-of-the-art (DCCAE) on average score. Best results are obtained using CorrNet-500-384. This experiment suggests that apart from multiview applications such as transfer learning, reconstructing missing view, and matching items across views, CorrNet can also be employed to exploit multiview data to improve the performance of a single-view task (such as monolingual bigram similarity).

Table 12: Spearman's Correlation for Bigram Similarity Data Set.

Model	AN	VN	Average
Baseline (LSI)	45.0	39.1	42.1
CCA	46.6	37.7	42.2
SplitAE	47.0	45.0	46.0
CorrAE	43.0	42.0	42.5
DistAE	43.6	39.4	41.5
FKCCA	46.4	42.9	44.7
NKCCA	44.3	39.5	41.9
DCCA	48.5	42.5	4.5
DCCAE	**49.1**	43.2	46.2
CorrNet	46.2	**47.4**	**46.8**

Notes: Results for other models are taken from Wang et al. (2015). The best results are in bold.

10 Conclusion and Future Work

In this article, we proposed correlational neural networks[5] as a method for learning common representations for two views of the data. The proposed model has the capability of reconstructing 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: cross-language document classification and transliteration equivalence detection. For both 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.

Acknowledgments

We thank Alexander Klementiev and Ivan Titov for providing the code for the classifier and data indices for the cross-language document classification task. We thank Janarthanan Rajendran for valuable discussions.

[5]Code for the model is available at https://github.com/apsarath/CorrNet.

References

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

Andrew, G., Arora, R., Bilmes, J., & Livescu, K. (2013). Deep canonical correlation analysis. In *Proceedings of the International Conference on Machine Learning*. JMLR.org

Arora, R., & Livescu, K. (2012). Kernel CCA for multi-view learning of acoustic features using articulatory measurements. In *Proceedings of the 2012 Symposium on Machine Learning in Speech and Language Processing* (pp. 34–37).

Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., . . . Bengio, Y. (2010). Theano: A CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference*.

Bird Steven, E. L., & Klein, E. (2009). *Natural language processing with Python*. Cambridge, MA: OReilly Media.

Chandar, S., Lauly, S., Larochelle, H., Khapra, M. M., Ravindran, B., Raykar, V. C., & Saha, A. (2013). *Multilingual deep learning*. Presented at NIPS Deep Learning Workshop, Lake Tahoe, CA.

Chandar, S., Lauly, S., Larochelle, H., Khapra, M. M., Ravindran, B., Raykar, V., & Saha, A. (2014). An autoencoder approach to learning bilingual word representations. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, & K. Q. Weinberger (Eds.), *Advances in neural information proceesing systems, 27* (pp. 1853–1861). Red Hook, NY: Curran.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, *12*, 2493–2537.

Cruz-Cano, R., & Lee, M.-L. T. (2014). Fast regularized canonical correlation analysis. *Computational Statistics and Data Analysis 70*, 88–100.

Dauphin, Y., Glorot, X., & Bengio, Y. (2011). Large-scale learning of embeddings with reconstruction sampling. In *Proceedings of the 28th International Conference on Machine Learning* (pp. 945–952). Madison, WI: Omnipress.

Dhillon, P., Foster, D., & Ungar, L. (2011). Multi-view learning of word embeddings via CCA. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems, 24* (pp. 199–207). Red Hook, NY: Curran.

Gao, J., He, X., Yih, W.-t., & Deng, L. (2014). Learning continuous phrase representations for translation modeling. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics* (vol. 1, pp. 699–709). Stroudsburg, PA: Association for Computational Linguistics.

Hardoon, D. R., Szedmak, S., & Shawe-Taylor, J. (2004). Canonical correlation analysis: An overview with application to learning methods. *Neural Computation*, *16*(12), 2639–2664.

Hermann, K. M., & Blunsom, P. (2014a). Multilingual distributed representations without word alignment. In *Proceedings of International Conference on Learning Representations*. arXiv.

Hermann, K. M., & Blunsom, P. (2014b). Multilingual models for compositional distributed semantics. In *Proceedings of the 52nd Annual Meeting of the Association* *for Computational Linguistics* (vol. 1, pp. 58–68). Stroudsburg, PA: Association for Computational Linguistics.

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

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

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*. Abs/1502.03167.

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

Klementiev, A., Titov, I., & Bhattarai, B. (2012). Inducing crosslingual distributed representations of words. In *Proceedings of the International Conference on Computational Linguistics*.

Koehn, P. (2005). Europarl: A parallel corpus for statistical machine translation. In *Proceedings of the Machine Translation Summit*.

Kumaran, A., Khapra, M. M., & Li, H. (2010). Report of news 2010 transliteration mining shared task. In *Proceedings of the 2010 Named Entities Workshop* (pp. 21–28). Stroudsburg, PA: Association for Computational Linguistics.

Li, H., Kumaran, A., Zhang, M., & Pervouvhine, V. (2009). Whitepaper of news 2009 machine transliteration shared task. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration* (pp. 19–26). Stroudsburg, PA: Association for Computational Linguistics.

Lu, A., Wang, W., Bansal, M., Gimpel, K., & Livescu, K. (2015). Deep multilingual correlation for improved word embeddings. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Stroudsburg, PA: Association for Computational Linguistics.

Lu, Y., & Foster, D. P. (2014). large scale canonical correlation analysis with iterative least squares. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems, 27* (pp. 91–99). Red Hook, NY: Curran.

Luo, Y., Tao, D., Wen, Y., Ramamohanarao, K., & Xu, C. (2015). *Tensor canonical correlation analysis for multi-view dimension reduction*. arXiv:1502.02330.

Mikolov, T., Le, Q., & Sutskever, I. (2013). *Exploiting similarities among languages for machine translation* (Technical report). arXiv.

Mishra, S. (2009). Representation-constrained canonical correlation analysis: A hybridization of canonical correlation and principal component analyses. *Journal of Applied Economic Sciences*, *4*, 115–124.

Mitchell, J., & Lapata, M. (2010). Composition in distributional models of semantics. *Cognitive Science*, *34*(8), 1388–1429.

Ngiam, J., Khosla, A., Kim, M., Nam, J., Lee, H., & Andrew, N. (2011). Multimodal deep learning. In *Proceedings of the International Conference on Machine Learning*. JMLR.org

Nielsen, F. Å., Hansen, L. K., & Strother, S. C. (1998). Canonical ridge analysis with ridge parameter optimization. *NeuroImage*, *7*, 5758.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Duchesnay, E. (2011). SCIKIT-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Soyer, H., Stenetorp, P., & Aizawa, A. (2015). Leveraging monolingual data for crosslingual compositional word representations. In *Proceedings of the 3rd International Conference on Learning Representations*. New York: ACM.

Tenenhaus, A., & Tenenhaus, M. (2011). Regularized generalized canonical correlation analysis. *Psychometrika*, *76*(2), 257–284.

Udupa, R., & Khapra, M. M. (2010). Transliteration equivalence using canonical correlation analysis. In *Proceedings of the 32nd European Conference on IR Research* (pp. 75–86). New York: Springer.

Vinod, H. (1976). Canonical ridge and econometrics of joint production. *Journal of Econometrics*, *4*(2), 147–166.

Wang, W., Arora, R., Livescu, K., & Bilmes, J. (2015). On deep multi-view representation learning. In *Proceedings of the International Conference on Machine Learning*. JMLR.org

Zou, W. Y., Socher, R., Cer, D., & Manning, C. D. (2013). Bilingual word embeddings for phrase-based machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. New York: ACM.

Received April 27, 2015; accepted September 20, 2015.