Linking Image and Text with 2-Way Nets

Aviv Eisenschtat[1] and Lior Wolf[1,2]

[1]The Blavatnik School of Computer Science, Tel Aviv University, Israel
[2]Facebook AI Research

Abstract

Linking two data sources is a basic building block in numerous computer vision problems. Canonical Correlation Analysis (CCA) achieves this by utilizing a linear optimizer in order to maximize the correlation between the two views. Recent work makes use of non-linear models, including deep learning techniques, that optimize the CCA loss in some feature space. In this paper, we introduce a novel, bi-directional neural network architecture for the task of matching vectors from two data sources. Our approach employs two tied neural network channels that project the two views into a common, maximally correlated space using the Euclidean loss. We show a direct link between the correlation-based loss and Euclidean loss, enabling the use of Euclidean loss for correlation maximization. To overcome common Euclidean regression optimization problems, we modify well-known techniques to our problem, including batch normalization and dropout. We show state of the art results on a number of computer vision matching tasks including MNIST image matching and sentence-image matching on the Flickr8k, Flickr30k and COCO datasets.

1. Introduction

Computer vision emerged from its roots in image processing when researchers began to seek an understanding of the scene behind the image. Linking visual data X with an external data source Y is, therefore, the defining task of computer vision. When applying machine learning tools to solve such tasks, we often consider the outside source Y to be univariate. A more general scenario is the one in which Y is also multidimensional. Examples of such view to view linking include matching a video and concurrent audio, matching an image with its textual description, matching images from two fixed views, etc.

The classical method of matching vectors between two different domains is Canonical Correlation Analysis (CCA). The algorithm has been generalized in many ways: regularization was added [29], kernels were introduced [2, 30, 4], versions for more than two sources were developed [41] etc. Recently, with the advent of deep learning methods, deep versions were created and showed promise.

The current deep CCA methods optimize the CCA loss on top of a deep neural network architecture. In this work, an alternative is presented in which a network is built to map one source X to another source Y and back. This architecture, which bears similarities to the encoder-decoder framework [11], employs the Euclidean loss.

The Euclidean loss is hard to optimize for, when compared to classification losses such as the cross entropy loss. We, therefore, introduce a number of contributions that are critical to the success of our methods. These include: (i) a mid-way loss term that helps support the training of the hidden layers; (ii) a decorrelation regularization term that links the problem back to CCA; (iii) modified batch normalization layers; (iv) a regularization of the scale parameter that ensures that the variance does not diminish from one layer to the next; (v) a tied dropout method; and (vi) a method for dealing with high-dimensional data.

Taken together, we are able to present a general and robust method. In an extensive set of experiments, we present clear advantages over both the classical and recent methods.

2. Previous work

Canonical Correlation Analysis (CCA) [14] is a statistical method for computing a linear projection for two views into a common space which maximizes their correlation. CCA plays a crucial role in many computer vision applications including multiview analysis [1], multimodal human behavior analysis [39], action recognition [16], and linking text with images [18]. There are a large number of CCA variants including: regularized CCA [44], Nonparametric canonical correlation analysis (NCCA) [31], and Kernel canonical correlation analysis (KCCA) [2, 30, 4], a method for producing non-linear, non-parametric projections using the kernel trick. Recently, randomized non-linear component analysis (RCCA) [32] emerged as a low-rank approxi-

mation of KCCA. While CCA is restricted to linear projections, KCCA is restricted to a fixed kernel. Both methods do not scale well with the size of the dataset and the size of the representations. A number of methods [3, 46, 6, 34] based on Deep Learning were recently proposed that aim to overcome these drawbacks. Deep canonical correlation analysis [3] processes the pairs of inputs through two network pipelines and compares the results of each pipeline via the CCA loss.

[48] and [45] extend [3] to the task of images and text matching. The first employs the same model and training process of [3] while the latter employs a different training scheme on the same architecture. Unlike [48] and [45] we present a novel deep model for matching images and text.

Other deep CCA methods, including ours, are inspired by a family of encoding/decoding unsupervised generative models [12, 5, 28, 42, 43] that aim to capture a meaningful representation of input x by applying a non-linear encoding function $E(x)$, decoding the encoded signal using a non-linear decoding function $D(x)$ and minimizing the squared L2 distance between the original input and the decoded output. Some of the auto-encoder based algorithms incorporate a noise on the input [42, 43] or enforce a desired property using a regularization term [28].

Correlation Networks (CorrNet) [6] and Deep canonically correlated autoencoders (DCCAE) [46] expand the auto-encoder scheme by considering two input views and two output views. The encoding is shared between the two views (CorrNet) or the differences in the encodings are minimized (DCCAE). In both cases, it serves as a common bottleneck. Our model goes from one view to the other (in both directions) and not from each view to a reconstructed view.

The CCA loss is used by both CorrNet and DCCAE. The latter contribution explicitly states that the L2 loss is inferior to the CCA loss term [46]. Our network, however, uses L2 successfully. This reinforces the need to apply the methods we propose in this work in order to enable effective training based on the L2 loss. For this end, we introduce innovative techniques based on common practices in deep learning, adapted to the problem at hand. These techniques include: dropout, batch normalization, and leaky ReLUs. While the latter is applied as is, the former two need to be carefully modified for our networks.

Dropout [40] is a regularization method developed to reduce over-fitting in deep neural networks by zeroing a group of neurons at each training iteration. This stochastic elimination reduces the co-adaptation between neurons in the same layer and simulates the training of an ensemble of networks with shared weights.

Batch Normalization [37] is used as a stabilizing mechanism for training a neural network by scaling the output of a hidden layer to zero norm and unit variance. This scaling lowers the change of distribution between neurons throughout the network and helps to speed up the training process.

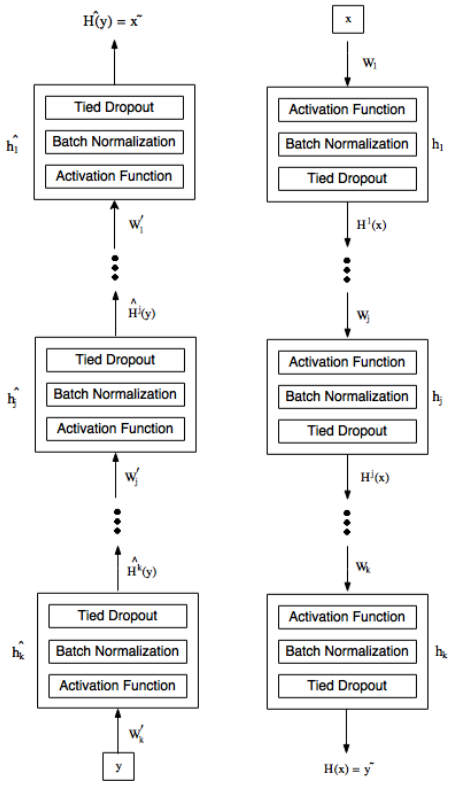

Figure 1: The 2-way network model. Each channel transforms one view into the other. A middle representation is extracted for correlation maximization

Rectified Linear Unit (ReLU) [33] is a non-linear activation function that does not suffer from the saturation phenomenon, which the classical sigmoids suffer from. Conventional ReLU zero negative activations, and as a result, no gradient is produced for many of the neurons. A few variants of ReLU were, therefore, proposed [26, 9] that reduce the effect of negative activations, but do not zero them completely. Similar to [26] and unlike [9], we do not train the leakiness parameter and instead set it to a constant value.

As one of our contributions, we add a regularization term that removes the pairwise covariances of the learned features. A similar term was recently reported in work [7] as part of a classification system (unrelated to modeling correlations between vectors). We adapt their terminology when describing our bi-directional term.

3. The Network Model

This section contains a detailed description of our proposed model, which we term the 2-way net[1] . The model

[1]Code can be found at https://github.com/aviveise/2WayNet

utilizes the L2 loss in order to create a bi-directional mapping between two vector spaces. The absence of a correlation based loss (such as in DeepCCA [3] and CorrNet [6]) makes this model simpler. Like other regression problems, there are inherent challenges in obtaining meaningful solutions [8]. These challenges are further amplified by the multivariate and layered structure of the performed regression. We, therefore, modify the problem in various ways, each contributing to the overall success.

3.1. Basic Architecture

Our proposed architecture is illustrated in Fig. 1. It contains two reconstruction channels. Both channels contains k hidden layers $\{h_1, h_2, ..., h_k\}$ and $\{\hat{h_1}, \hat{h_2}, ..., \hat{h_k}\}$. Lets define $H_i(x)$ and $\hat{H_i}(y)$ as the output of each channel at layer i given network inputs x and y respectively, the model is optimized to minimize the Eucledean loss between both $\hat{H_i}(y)$ and x, and $H_i(x)$ and y. The two channels share weights and dropout function as explained in 3.5

The activations of each hidden layer are computed by a function $h(x) = \Phi(Wx + b_2)$ from $\mathbb{R}^{d_1}$ to $\mathbb{R}^{d_2}$, where $W \in \mathbb{R}^{d_2 \times d_1}$ is the weight matrix, $b_2 \in \mathbb{R}^{d_2}$ is the bias vector and Φ is a non-linear function, which in our model is a leaky rectified linear unit [26]. The tied layer is given as $\hat{h}(y) = \Phi\left(W^T y + b_1\right)$, and employs the transpose of the matrix W and an untied bias term $b_1 \in \mathbb{R}^{d_1}$.

Given a pair of views $x \in \mathbb{R}^{d_x}$ and $y \in \mathbb{R}^{d_y}$, two reconstructions are created: $\tilde{x} \in \mathbb{R}^{d_x}$ and $\tilde{y} \in \mathbb{R}^{d_y}$ by employing the two networks $H = h_1 \circ h_2 \circ ... \circ h_k$ and $\hat{H} = \hat{h}_k \circ \hat{h}_{x_{k-1}} \circ ... \circ \hat{h}_1$, as $\tilde{x} = \hat{H}(y)$ and $\tilde{y} = H(x)$.

Loss is measured between x and $\tilde{x}$ and y and $\tilde{y}$. Moreover, the Euclidean distance is also minimized directly on the desired representations. In order to do so, we select a mid-network position $j = \lceil k/2 \rceil$. We then add a loss term by considering the two networks: $H^j = h_1 \circ h_2 \circ ... \circ h_j$, and $\hat{H}^j = \hat{h}_k \circ \hat{h}_{x_{k-1}} \circ ... \circ \hat{h}_{j+1}$. A loss term is then added that compares $H^j(x)$ and $\hat{H}^j(y)$.

The overall loss is given by the three terms $L_x = \|x - \tilde{x}\|^2$, $L_y = \|y - \tilde{y}\|^2$, and $L_h = \|H^j(x) - \hat{H}^j(y)\|^2$. Note that minimizing Euclidean distances differs from maximizing the pairwise correlations as is done in CCA and its variants DeepCCA [3] and RCCA [32].

In our experiments, in order to compare with previous work, we use the correlation as the success metric. As the Lemma below shows, there is a connection between the correlation of two vectors and their Euclidean distance, this connection also depends on the variance of the vectors.

Lemma 1. *Let $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^n$ denote two paired lists of n matching samples from two random variables with zero mean and σ_x^2 and σ_y^2 variances. Then, the correlation between the two n dimensional samples x and y equals $\frac{\sigma_x}{2\sigma_y} + \frac{\sigma_y}{2\sigma_x} - \frac{\|x-y\|^2}{2n\sigma_x\sigma_y}$.*

Proofs for the lemmas presented in this paper are described in the appendix. Given a batch of samples from views x and y, we measure the correlation between the outputs of two matching layers, $\{h_j(x_1), ..., h_j(x_n)\}$ and $\{\hat{h}_j(y_i), ..., \hat{h}_j(y_n)\}$ as the sum of correlations between the activations of each matching neuron. The Lemma below extends Lemma 1 and shows that the sum of correlations which we aim to maximize is bounded by a function of the Euclidean loss between the two representations.

Lemma 2. *Given two matching hidden layers, h_j and $\hat{h_j}$ with m neurons each. a_k is the activation vector of neuron k from h_j with standard deviation σ_{a_k} and b_k is the activation vector of neuron k from $\hat{h}_j$ with standard deviation σ_{b_k}. Each vector is produced by feeding a batch of samples of size n from views x and y through channels H and $\hat{H}$ respectively. The sum of correlations C is bounded by:*

$$\sum_{k=1}^{m} C_k \geq \frac{1}{2}\sum_{k=1}^{m}\left(\frac{\sigma_{a_k}^2 + \sigma_{b_k}^2}{\sigma_{a_k}\sigma_{b_k}}\right) - \frac{1}{2n}\sum_{k=1}^{m}\|a_k - b_k\|^2 \sum_{k=1}^{m}\sigma_{a_k}^{-1}\sigma_{b_k}^{-1} \quad (1)$$

From the above Lemma, we can conclude that by minimizing the L2 loss together with maximizing the variance of each neuron activation will result in maximization of the sum of correlations.

Solving this regression problem tends to eliminate the variance of the output representations. To overcome this limitation, we add two instruments. The first is batch normalization layer [37] (BN) after each hidden layer. The settings of the batch normalization layer differ from the common settings to adapt to this model. Another instrument is regularizing the gamma parameter the batch normalization layer introduces. More details can be found below.

To the loss term, we add regularization terms. The first is weight decay $R_w = \sum \|W\|^2$. A second regularization term is added in order to reduce the cross correlations between the network activations of the same layer. The property we encourage is inherent to CCA-based solutions where decorrelation is enforced. In our network solutions, we add a soft regularization term. During training, we consider the N samples of a single batch $\{(x_i, y_i)\}_{i=1}^N$ and consider the set of mid-network activations $\{(H^j(x_i), \hat{H}^j(y_i))\}_{i=1}^N$. The decorrelation regularization term is given by:

$$R_{decov} = \frac{1}{2}\left(\|C_h\|_F^2 - \|diag\left(C_h\right)\|_2^2\right) + \frac{1}{2}\left(\|C_{\hat{h}}\|_F^2 - \|diag\left(C_{\hat{h}}\right)\|_2^2\right), \quad (2)$$

where $C_h = \frac{1}{N}\sum_i H^j(x_i)^\top H^j(x_i)$ is the covariance estimator for $H^j(x)$ and $C_{\hat{h}} = \frac{1}{N}\sum_i \hat{H}^j(y_i)^\top \hat{H}^j(y_i)$ is the

covariance estimator for $\hat{H}^j(y)$. This regularization term is minimized when the off-diagonal coefficients of both C_h and $C_{\hat{h}}$ are zero.

3.2. Batch normalization layers

As shown above, in order to maximize the correlation we need not only to minimize the Euclidean loss but also to increase the variance of each neuron's output. This is done by introducing a batch normalization layer [37] customized to meet the model's needs.

Given a vector of activations $a = [a_1, \ldots, a_d]$ produced by one of the network's hidden layers for a given batch of inputs, we normalize a to produce $a' = [a'_1, \ldots, a'_d]$, where $a'_k = \frac{a_k - \mu_k}{\sigma_k}$ and μ_k and σ_k^2 are the mean and variance of neuron k on the given batch. This is followed by scaling and shifting by learned parameters to produce $a''_k = \gamma_k a'_k + \beta_k$. The BN layer mitigates the loss of variance by enforcing unit variance and by removing the influence of the weights of the hidden layer on the output's variance.

BN layers are usually placed before the non-linearity or on the input of the layer as a preprocessing phase as shown in [10]. This setting poses several problems. First, ReLU lowers the variance of the output which is counterproductive to our goal. Second, applying ReLU after BN has the effect of zeroing every k when a_k is below the mean in a given batch plus the term β_k/γ_k. Typically, β_k is initialized to zero and for a symmetric activation distribution, half of the activations are zeroed. When employing a bi-directional network, the zeroing effect occurs in both directions.

In order to estimate the magnitude of this effect, let us assume that we have a process that at time i outputs two vectors $u_i = H^j(x_i)$ and $v_i = \hat{H}^j(y_i)$, both in $\mathbb{R}^d$, which are the hidden representation at layer j for a pair of samples (x_i, y_i). Denote by ρ_k the correlation between the activations at neuron k.

Let $s_i = \{k | u_i(k) > \mu_k\}$ be the group of indices of the values in u_i that are larger than their population mean. Let $\hat{s}_i = \{k | v_i(k) > \hat{\mu}_k\}$ be the equivalent for the vectors v_i. We observe the intersection $s_i \cap \hat{s}_i$, which is the group of active neurons, following a threshold at the mean value on both u_i and v_i. As the Lemma below shows, even if the correlation ρ_k is relatively high, the size of the intersection set $s_i \cap \hat{s}_i$ is closer to the value $d/4$ obtained for randomly permuted vectors than to the maximal value of $d/2$.

Lemma 3. *Assume that u_i and v_i are drawn from a multivariate normal distribution with zero mean and the identity covariance matrix, such that the correlation between $u_i(k)$ and $v_i(k)$ for all k is $\rho_k = \rho$. Then, $E(|s_i \cap \hat{s}_i|) = d\left[\frac{1}{4} + \frac{\sin^{-1}\rho}{2\pi}\right]$.*

Even in the case of a correlation as high as 0.6, the intersection will include only about 35% of the neurons. For neurons k not in this intersection, either both sides $u_i(k)$ and $v_i(k)$ are zero, meaning that no backpropagation occurs, or only one neuron is active, in which case only that side is updated and the update is a simple shrinking effect, since the loss is the magnitude of the activation.

In order to break this symmetry, we choose to employ the BN after the non-linearity. This allows the network to choose weights that result in mostly positive activations, which remain positive after the ReLU activation units.

3.3. Highly leaky ReLU

Another method to prevent the harmful effects of zeroing is by using leaky ReLU as our non-linear function. Leaky ReLU was first introduced by [26] in order to overcome the difficulties that arise from the elimination of the gradients from neurons with negative activation. In the 2-Way network, this effect is amplified, and we find leaky ReLU units to be extremely important. Formally, a leaky ReLU is defined as:

$$y_i = \begin{cases} x_i & \text{if } x \geq 0 \\ ax_i & \text{if } x < 0 \end{cases}$$

where $a < 1$ is the leakiness coefficient and is fixed during both training and testing. In all of our experiments, we use a leakiness coefficient of 0.3. This value was selected on the validation set of the Flickr8k experiment described in Section 4 and is used for all experiments.

Using leaky ReLU helps to reduce the effect discussed in Section 3.2 but does not replace the need for performing BN after the non-linearity. As Lemma 3 shows, more than half of the neurons will be multiplied by the leakiness coefficient while their matching neuron will not. This asymmetric scaling adds an artificial distance between the matching neurons, which, in turn, increases the L2 loss and reduces the training efficiency.

3.4. Variance injection

Applying BN on the output of each hidden layer is not enough. The variance can still vanish during training. The problem is that the γ factor introduced by each BN layer can be arbitrary and can diminish during training, resulting in low variance. To encourage high variance, we introduce a novel regularization term of the form $R_\gamma = \sum_{j,k}(1/\gamma_{jk})^2$, where γ_{jk} is the scaling parameter for neuron k in layer j.

This regularization term is enough to force the network to avoid solutions with low variance and to seek more informative output. This is demonstrated experimentally in the ablation study of Section 4.

The compound loss term we employ is of the form:

$$L = L_x + L_y + L_h + \lambda_w R_w + \lambda_{decov} R_{decov} + \lambda_\gamma R_\gamma$$

Where λ_w,λ_{decov}, and λ_γ are the regularization coefficients. While it seems that three regularization tradeoff hyperparameters would make selecting the parameter values diffi-

cult, the converse is true: in all of our varied set of experiments $\lambda_\gamma = \lambda_w$, and λ_{decov} is either set to a very high value of $1/2$ or, for small datasets, to $1/20$ (see Section 4). Moreover, by adding these terms, the network is much less sensitive to the selection of λ_w and allows us to learn with a much higher learning rate.

3.5. Tied dropout

Dropout [40] is a form of regularization method that simulates the training of multiple networks with shared weights. Dropout zeros neurons by element-wise multiplying the output of a hidden layer consisting of d neurons for a batch of n samples with a random matrix B of size $d \times n$. Each element of B is drawn independently from a Bernoulli distribution with a parameter p. Since dropout eliminates random neurons, it prevents co-adaptation of neurons, which is a desirable property for correlation analysis. However, using dropout, as is, in our proposed model is harmful. This is because the 2-Way network aims to enhance correlations between parallel layers h^j and $\hat{h}^j$. The elimination of neurons independently in the hidden layers creates an artificial loss, even for a perfect matching.

Let p be the dropout parameter for layer j, assume that the same parameter is applied on both directions. In probability $(1-p)^2$, a pair of matching neurons is active on both sides and learning occurs with the true gradient. In probability p^2, the pair of matching neurons is silent on both sides and no learning occurs. In probability $2p(1-p)$, only one neuron is active resulting in a shrinking effect on the other neuron. Here, too, shrinking of activations is can be damaging since it might lead to a state of constant representation.

For a dropout probability of $p = 0.5$, half of the gradients would stem from a match which is silent on exactly one side, and the harmful effect is clearly seen in Section 4. To overcome this problem, we introduce a tied dropout layer, in which the same random matrix B^j is applied to pairs of matching hidden layers: h^j and $\hat{h}^j$, $j = 1..K$. This sharing eliminates the artifacts introduced by the conventional dropout while preserving the benefits of the stochastic process and helps avoid over-fitting.

Using tied dropout layer changes the distribution of the activations. In order to match the distribution at test time, we incorporate a scaling factor at train time. Assume that the activations of a single neuron are zero-centered. As discussed below, most post BN activations are almost exactly centered. In this case, the variance of the neuron activations is simply the sum of the squared activations. During training, only a ratio $1-p$ of the activations contributes to the variance. Therefore, we divide the activations by $\sqrt{1-p}$.

3.6. Training high dimensional inputs

Some of the experiments shown below contain high dimensional data. High dimensional input directly increases the number of parameters and can cause over-fitting as well as an increase in training time and memory usage. To lower the number of parameters, we introduce a new type of layer we term locally dense layer. Such layer of size n is composed of m different dense layers $\bar{h_1}, ..., \bar{h_m}$ of size $\frac{n}{m}$ each. Input x of size d_x is divided into m different parts of size $\frac{d_x}{m}$ and each part x_i is connected into one of the dense layers $\bar{h_i}$. The outputs of all inner hidden layers are concatenated, thus producing the locally dense layer's output. To the output, we add a regular bias term b of size n. Using this layer reduces the number of parameters by a factor of m comparing to a conventional dense layer. In the experiments below, when dealing with high dimensional input, we use a locally dense layer with two inner dense layers.

4. Experiments

We first present a detailed analysis of the two datasets most commonly used in the literature for examining recent CCA variants: MNIST half matching and X-Ray Microbeam Speech data (XRMB). We then provide additional experiments on the problem of image to sentence matching, showing state of the art results on the Flickr8k, Flickr30k and COCO datasets.

4.1. Comparison with published results

We follow the conventional way of evaluating the performance of CCA variants and compute the sum of the correlations of the top c shared (canonical) representation variables found. The datasets used for this comparison are MNIST and XRMB. In both MNIST and XRMB experiments, we set $\lambda_{decov} = \lambda_W = \lambda_\gamma = 0.05$. For training, we used stochastic gradient descent with a learning rate of 0.0001 which was halved every 20 epochs. A momentum of 0.9 is used and a tied dropout probability of 0.5.

MNIST half matching The MNIST handwritten digits dataset [19] contains 60,000 images of handwritten digits for training and 10,000 images for testing. Each image is cut vertically into two halves, resulting in 392 features each. The goal is to maximize the correlation of the top $c = 50$ canonical variables. The model used is composed of three layers of size 392, 50 and 392 respectively, noted as 392-50-392. The middle layer was taken as the output.

X-Ray Microbeam Speech data The XRMB [47] dataset contains simultaneous acoustic and articulatory recordings. The articulatory data is represented as a 112 dimensional vector. The acoustic data are the MFCCs [24] for the same frames, yielding a 273 dimensional vector at each point in time. For benchmarking, 30,000 random samples are used for training, 10,000 for cross-validation and 10,000 for testing. The correlation is measured across the $c = 112$ top correlated canonical variables. The same training configuration of the MNIST experiment was used for the XRMB

dataset. For XRMB, we tested our model using hidden layer configuration of 560-280-112-680-1365.

Tab. 1 contains correlation comparisons on the MNIST and XRMB datasets of six CCA variants besides our proposed method. As can be seen, our method ("2WayNet") outperforms all literature methods by a large margin on the XRMB dataset. On the MNIST dataset, in which the literature results are closer to the maximal value of 50, our method is able to regain half of the remaining correlation.

Method	MNIST	XRMB
Regularized CCA [44]	28.0	16.9
DCCA [3]	39.7	92.9
RCCA [32]	44.5	104.5
DCCAE [46]	25.34	41.47
CorrNet [6]	48.07	95.01
NCCA [31]	NA	107.9
2WayNet	**49.15**	**110.18**

Table 1: Comparison between various methods on the XRMB and MNIST datasets. The reported values are the sum of the correlations between the learned representations of the two views. Following the literature, in these benchmarks MNIST employs a 50D shared representation space, and XRMB a 112D one.

4.2. Image annotation and search

We next evaluate the proposed model on the sentence-image matching task. In this task, each dataset contains a set of images and five matching sentences per image. For each dataset, we test our model on two tasks, searching an image given a query sentence and matching a sentence given an image. We measure our performance on three datasets, Flickr8k [13], Flickr30k [49] and COCO [22], each containing 8,000, 30,000 and 123,000 images respectively.

Images are presented by the representation layer of the VGG network [38] as vectors of size 4096. Sentences are represented using the published code of [18]. Among the available text encodings, we employ the concatenation of the Fisher Vector encoding (GMM) and the Fisher Vector of the HGLMM distribution introduced in [18]. Each sentence is thus represented as a 36,000D vector. Going from the image to the much larger sentence representation, we trained networks containing two conventional hidden layers of sizes 2000 and 3000 and an additional locally dense layer of 16000 neurons and $m = 2$ for Flickr30k and COCO datasets. For Flickr8k, due to the relatively small dataset, we used a dense layer of 4000 neurons. Correlation is used as a similarity measure between images and sentences. To this end we use the middle network output from each channels, resulting in a representation vector of size 3000.

The Flickr8k dataset is provided with training, validation, and test splits. For Flickr30K and COCO, no splits are given, and we use the same splits used by [18]. λ_{deconv} is set to a value of $1/2$, which almost eliminated all off-diagonal covariances at the middle layer. The other parameters are set as in the MNIST and XRMB experiments.

Tab. 2 compare our results to the state-of-the-art methods on the image-sentence matching task. We also report results that we computed for the RCCA method [32]. The open implementations of the various deep CCA methods do not seem to scale well enough for this benchmark. Our proposed method achieves best performance almost across all scores, especially in the image annotation task, where we improved by a large margin for the three datasets, and especially when considering the top result (r@1).

4.3. Ablation analysis

We perform an ablation analysis aimed at isolating the effect of the various architectural novelties suggested. Experiments were conducted on the Flickr8k, Flickr30k, MNIST and XRMB datasets. Each experiment uses the baseline configuration with only one alternation.

Batch Normalization For this experiment, we used different settings for the BN layer. The configuration settings include: (1) without BN, (2) with conventional BN (before ReLU) without regularizing γ, (3) with post-ReLU BN, without regularizing γ, (4) using BN before the ReLU with $\lambda_\gamma = 0.05$, and (5) our proposed method: BN applied only after ReLU with $\lambda_\gamma = 0.05$. Tab. 3 report the performance of the various configurations in terms of correlation and the mean variance of all features on the validation set.

As Tab. 3 shows, batch normalization has a profound effect on the network's results. Results taken without batch normalization were trained with lower learning rate, using higher learning rate prevented the training from converging. We can also see that using the $1/\gamma$ regularization term significantly increases the variance of the hidden representation, which, in turn, stabilizes the training process and improves correlation. The effect studied in Section 3.2 is clearly visible in the ablation study, positioning the BN layer after the Leaky ReLU prevents an unbalance representations as can be seen by the difference in variances, which increases the correlation of two representation significantly. Tab. 4 contains r@1 results for the same experiments on the Flickr8k dataset. As in Tab. 3 our suggested configuration achieves the base recall rates.

Tied Dropout We trained the same base configuration described above. We also tested our proposed method using a conventional dropout and without dropout. In all experiments, the dropout probability p was set at 0.5.

As can be seen, the performance drops when using the conventional dropout instead of the proposed tied dropout layer. The benefits of the tied dropout layer are most significant on the large datasets Flickr8k and Flickr30k, where over-fitting is likely. The shrinking effect discussed in Sec-

Model	Flickr8k				Flickr30k				COCO			
	Search		Annotate		Search		Annotate		Search		Annotate	
	r@1	r@5	r@1	r@5	r@1	r@5	r@1	r@5	r@1	r@5	r@1	r@5
NIC [35]	19.0	NA	20.0	NA	17.0	NA	17.0	NA	NA	NA	NA	NA
SC-NLM [17]	12.5	37.0	18.0	40.9	16.8	42.0	23.0	50.7	NA	NA	NA	NA
m-RNN [27]	11.5	31.0	14.5	37.2	22.8	50.7	35.4	63.8	29.0	42.2	41.0	73.0
m-CNN [25]	20.3	47.6	24.8	53.7	26.2	56.3	33.6	64.1	32.6	68.6	42.8	73.1
DCCA [48]	12.7	31.2	17.9	40.3	12.6	31.0	16.7	39.3	NA	NA	NA	NA
BRNN [15]	NA	NA	NA	NA	15.2	37.7	22.2	48.2	27.4	60.2	38.4	69.9
RNN-FV [20]	23.2	**53.3**	31.6	61.2	27.4	55.9	35.9	62.5	30.2	65.0	40.9	75.0
VQA-A [23]	17.2	42.8	24.3	52.2	24.9	52.6	33.9	62.5	37.0	70.9	50.5	**80.1**
NLBD [45]	NA	NA	NA	NA	29.7	**60.1**	40.3	**68.9**	39.6	**75.2**	50.1	79.7
CCA [18]	21.3	50.1	31.0	59.3	23.5	52.8	35.0	62.1	25.1	59.8	39.4	67.9
RCCA [32]	18.7	31.1	11.7	19.2	22.7	34.2	28.3	48.2	NA	NA	NA	NA
2WayNet	**29.3**	49.7	**43.4**	**63.2**	**36.0**	55.6	**49.8**	67.5	**39.7**	63.3	**55.8**	75.2

Table 2: The recall rates for the Flickr8k, Flickr30k and COCO image to sentence matching benchmarks. In image search, we show the percent of correct matches for the top retrieval out of all test images (r@1 for search). In image annotation, given a query image, fetching one of five matching sentences is considered a success. Recall rates for the top five (r@5) denote the cases in which a successful match exists in any of the top five results. The experiments reported for regularized CCA, RCCA, and our 2-way net all use the same sentence and image representation. Sentences are represented as the concatenation of the GMM-FV and the HGLMM-FV representations of [18]. . Images are represented as in [18].

Scenario	Flickr8k			Flickr30k			MNIST			XRMB		
	Corr	Var x	Var y	Corr	Var x	Var y	Corr	Var x	Var y	Corr	Var x	Var y
Suggested method	**1758**	0.65	0.64	**2135**	0.41	0.43	**49.15**	1.32	1.27	**110.18**	1.08	1.06
No BN	1482	1.90	1.71	1562	1.38	1.40	13.14	0	0	25.58	0	0
before ReLU, $\lambda_\gamma = 0$	1313	0.66	0.44	1385	0.37	0.28	48.40	0.18	0.18	107.55	0.15	0.15
after ReLU, $\lambda_\gamma = 0$	1598	1.34	1.25	1655	0.73	0.74	48.98	0.38	0.37	109.42	0.40	0.39
before ReLU, $\lambda_\gamma > 0$	1423	0.33	0.21	1322	1.80	0.96	48.76	0.73	0.72	108.79	0.50	0.50
No Dropout	1091	0.34	0.33	1446	0.57	0.52	49.00	1.33	1.33	109.69	0.79	0.79
Conventional dropout	1557	0.17	0.17	1658	0.12	0.14	48.77	1.90	1.90	93.24	0.24	0.16

Table 3: Ablation study on the Flickr8k, Flickr30k, MNIST and XRMB datasets, testing various batch normalization (BN), variance regularization and dropout options. We measure the variance in both views, X and Y (averaging the variance of all dimensions), and the obtained correlation. The suggested method is to apply BN only after ReLU with $\lambda_\gamma = 0.05$ and to employ tied dropout. All BN variants employ tied dropout with probability of 0.5. All dropout variants apply BN similarly to the suggested method.

Scenario	Search r@1	Annotate r@1
Suggested method	**29.3**	**43.4**
No BN	21.1	25.6
before ReLU, $\lambda_\gamma = 0$	26.9	39.6
after ReLU, $\lambda_\gamma = 0$	27.9	40.9
No Dropout	25.64	36.6
Conventional dropout	29.04	42.1

Table 4: Recall results on Flickr8k for the same experiments as described at Tab. 3.

tion 3.5 is clearly visible and is manifested as low variance of the output of the model based on conventional dropout, compared to a higher variance when using the tied dropout.

Leaky ReLU We also tested the contribution of other parameters on the model's performance. One of the major benefits was using leaky ReLU non-linearity. Using conventional ReLU resulted in large correlation loss of about 33% (1192 total correlation) for Flickr8k. **Loss terms** Another aspect we tested is the effect of various loss terms on correlation and recall rates. Removing L_h term results in a 31% (1230) decrease of correlation. This settles with Lemma 2 which links the output's correlation and L_h loss term. While the L_h loss increases the output's correlation, the reconstruction loss terms L_x and L_y decreases the result's correlation. Removing them both increases correlation by 56% (2752). While the correlation produced be-

tween the two views is higher without the two reconstruction losses, the dimensions of each representation are highly correlated resulting in a decrease of 87% in image search and 91% in image annotation performance as measured by recall@1: from the full method's performance of 29.3 and 43.4 for the tasks of image search and image annotation to 4.0 and 3.9 respectively. **Regularization** The effect for R_γ can be viewed in Tab. 3. Removing the R_decov results in a decrease of all measures. Image search r@1 and r@5 results decrease by 14% and 10% respectively and the image annotation r@1 and r@5 results decrease by 10% and 8% respectively. Moreover, the correlation is reduced by 4%. **Locally dense layer** To test the effect of the proposed locally dense layer, we trained our model on Flickr30k with a regular dense layer of the same size (16000 neurons) and with a regular dense layer of half the size. Image annotation r@1(r@5) results degrade by $7\%(3\%)$ and image search by $1\%(1\%)$ when using conventional 16000 neurons dense layer. Using dense layer half the size results in a drop of $13\%(9\%)$ for image annotation r@1(r@5) rates and $11\%(8\%)$ for image search recall rates r@1(r@5).

Parameter sensitivity: Fig. 2(a) shows the effect of different leakiness coefficient values on the correlation as measured on the validation sets of the MNIST and XRMB data sets. The results were obtained by training the network using leakiness coefficients ranging between 0 and 0.7. As can be seen, there is a large region of values that provide better performance than the conventional zero-leakiness ReLU. Fig. 2(b) shows the effect of the regularization weight λ_γ that controls the learned variance of the BN layer. The value used in our experiments seems to be beneficial and lies at a relatively wide high-performance plateau.

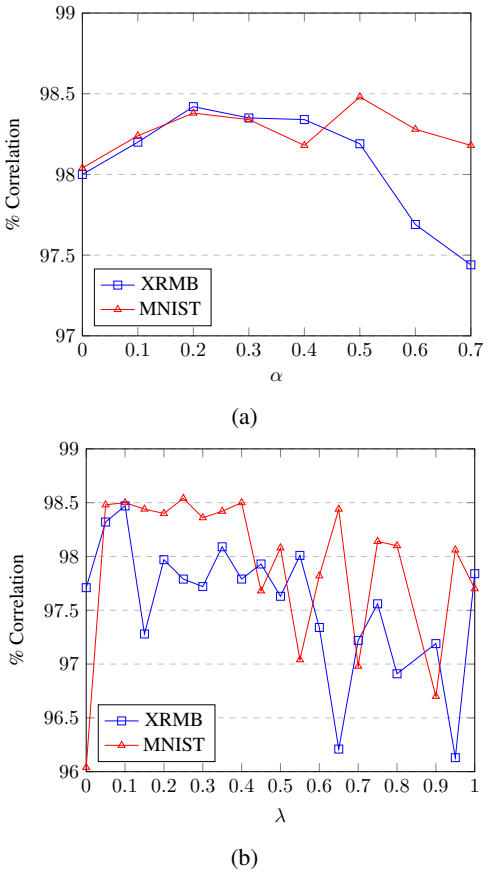

Figure 2: (a) The effect of the leakiness parameter on the MNIST and XRMB benchmarks, as measured on the validation set using the sum of correlations divided by the dimension (in percent). The solid red line depicts the MNIST results; the dashed black line depicts the XRMB results. (b) A similar plot showing the effect of coefficient λ_γ.

5. Conclusions

In this paper, we present a method for linking paired samples from two sources. The method significantly outperforms all literature methods in the highly applicable and well studied domain of correlation analysis, including the classical methods, their modern variants, and the recent deep correlation methods. We are unique in that we employ a tied 2-way architecture, reconstructing , and unlike most methods, we employ the Euclidean loss. In order to promote an effective training, we introduce a series of contributions that are aimed at maintaining the variance of the learned representations. Each of these modifications is provided with an analysis that explains its role and together they work hand in hand in order to provide the complete architecture, which is highly accurate.

Our method is generic and can be employed in any computer vision domain in which two data modalities are used. In addition, our contributions could also help in training univariate regression problems. In the literature, the Euclidean loss is often combined with other losses [36, 50], or replaced by an alternative loss [21] in order to mitigate the challenges of training regression problems. Our variance injection method can be easily incorporated into any existing network.

As future work, we would like to continue exploring the use of tied 2-Way networks for matching views from different domains. In almost all of our trained networks, the biases of the batch normalization layers in the solutions tend to have very low values. These biases can probably be eliminated altogether. In addition, in many encoder/decoder schemes, layers are added gradually during training. It is possible to adopt such a scheme to our framework, adding hidden layers in the middle of the network one by one.

Acknowledgments

This research is supported by the Intel Collaborative Research Institute for Computational Intelligence (ICRI-CI).

References

[1] Hal Daum III David W. Jacobs Abhishek Sharma, Kumar Abhishek. Generalized multiview analysis: A discriminative latent space. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[2] Shotaro Akaho. A kernel method for canonical correlation analysis. *arXiv preprint cs/0609071*, 2006.

[3] Galen Andrew, Raman Arora, Karen Livescu, and Jeff Bilmes. Deep canonical correlation analysis. In *International Conference on Machine Learning (ICML)*, 2013.

[4] Francis R Bach and Michael I Jordan. Kernel independent component analysis. *Journal of machine learning research (JMLR)*, 3(Jul):1–48, 2002.

[5] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Neural Information Processing Systems (NIPS)*, 2007.

[6] Sarath Chandar, Mitesh M. Khapra, Hugo Larochelle, and Balaraman Ravindran. Correlational neural networks. *Neural Computation*, 28(2):257–285, 2016.

[7] Michael Cogswell, Faruk Ahmed, Ross B. Girshick, Larry Zitnick, and Dhruv Batra. Reducing overfitting in deep networks by decorrelating representations. *arXiv preprint arXiv:1511.06068*, 2015.

[8] Li Fei-Fei and Andrej Karpathy. Stanford's cs231n class notes. http://cs231n.github.io/neural-networks-2/, 2015.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *International Conference on Computer Vision (ICCV)*, 2015.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. *arXiv preprint arXiv:1603.05027*, 2016.

[11] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[12] Geoffrey E. Hinton and Richard S. Zemel. Autoencoders, minimum description length and helmholtz free energy. In *Neural Information Processing Systems (NIPS)*, 1994.

[13] Micah Hodosh, Peter Young, and Julia Hockenmaier. Framing image description as a ranking task: Data, models and evaluation metrics. *Journal of Artificial Intelligence Research*, 47:853–899, 2013.

[14] Harold Hotelling. Relations between two sets of variates. *Biometrika*, 28(3/4):321–377, 1936.

[15] Andrej Karpathy and Fei-Fei Li. Deep visual-semantic alignments for generating image descriptions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[16] Tae-Kyun Kim and Roberto Cipolla. Canonical correlation analysis of video volume tensors for action categorization and detection. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 31(8):1415–1428, 2009.

[17] Ryan Kiros, Ruslan Salakhutdinov, and Richard S Zemel. Unifying visual-semantic embeddings with multimodal neural language models. *arXiv preprint arXiv:1411.2539*, 2014.

[18] Benjamin Klein, Guy Lev, Gil Sadeh, and Lior Wolf. Associating neural word embeddings with deep image representations using fisher vectors. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[19] Yann Lecun and Corinna Cortes. The MNIST database of handwritten digits. 1998.

[20] Guy Lev, Gil Sadeh, Benjamin Klein, and Lior Wolf. RNN fisher vectors for action recognition and image annotation. In *European Conference on Computer Vision (ECCV)*, 2016.

[21] Ofir Levy and Lior Wolf. Live repetition counting. In *International Conference on Computer Vision (ICCV)*, 2015.

[22] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollr, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision (ECCV)*, 2014.

[23] Xiao Lin and Devi Parikh. Leveraging visual question answering for image-caption ranking. *arXiv preprint arXiv:1605.01379*, 2016.

[24] Beth Logan et al. Mel frequency cepstral coefficients for music modeling. In *International Society for Music Information Retrieval (ISMIR)*, 2000.

[25] Lin Ma, Zhengdong Lu, and Lifeng Shang andHang Li. Multimodal convolutional neural networks for matching image and sentence. In *International Conference on Computer Vision (ICCV)*, 2015.

[26] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *International Conference on Machine Learning (ICML)*, 2013.

[27] Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, and Alan L. Yuille. Explain images with multimodal recurrent neural networks. *arXiv preprint arXiv:1410.1090*, 2014.

[28] Y-Lan Boureau Yann LeCun Marc'Aurelio Ranzato, Fu Jie Huang. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[29] Nick Martin and Hermine Maes. Multivariate analysis, 1979.

[30] Thomas Melzer, Michael Reiter, and Horst Bischof. Nonlinear feature extraction using generalized canonical correlation analysis. In *International Conference on Artificial Neural Networks (ICANN)*, 2001.

[31] Tomer Michaeli, Weiran Wang, and Karen Livescu. Nonparametric canonical correlation analysis. *arXiv preprint arXiv:1511.04839*, 2015.

[32] Paul Mineiro and Nikos Karampatziakis. A randomized algorithm for cca. *arXiv preprint arXiv:1411.3409*, 2014.

[33] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *International Conference on Machine Learning (ICML)*, 2010.

[34] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y. Ng. Multimodal deep learning. In *International Conference on Machine Learning (ICML)*, 2011.

[35] Samy Bengio Dumitru Erhan Oriol Vinyals, Alexander Toshev. Show and tell: A neural image caption generator. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[36] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Neural Information Processing Systems (NIPS)*, 2015.

[37] Christian Szegedy Sergey Ioffe. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, 2015.

[38] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[39] Yale Song, Louis-Philippe Morency, and Randall Davis. Multimodal human behavior analysis: Learning correlation and interaction across modalities. In *ACM International Conference on Multimodal Interaction (ICMI)*, 2012.

[40] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research (JMLR)*, 15(1):1929–1958, 2014.

[41] Arthur Tenenhaus and Michel Tenenhaus. Regularized generalized canonical correlation analysis. *Psychometrika*, 76(2):257, 2011.

[42] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *International Conference on Machine Learning (ICML)*, 2008.

[43] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research (JMLR)*, 11:3371–3408, 2010.

[44] H. D. Vinod. Canonical ridge and econometrics of joint production. *Journal of Econometrics*, 4(2):147–166, May 1976.

[45] Liwei Wang, Yin Li, and Svetlana Lazebnik. Learning deep structure-preserving image-text embeddings. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[46] Karen Livescu Jeff Bilmes Weiran Wang, Raman Arora. On deep multi-view representation learning. In *International Conference on Machine Learning (ICML)*, 2015.

[47] John R. Westbury. X-ray microbeam speech production database user's handbook. Technical report, University of Wisconsin, 1994.

[48] Fei Yan and Krystian Mikolajczyk. Deep correlation for matching images and text. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[49] Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78, 2014.

[50] Cong Zhang, Hongsheng Li, Xiaogang Wang, and Xiaokang Yang. Cross-scene crowd counting via deep convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.