Group Sparse CNNs for Question Classification with Answer Sets

Mingbo Ma **Liang Huang**
School of EECS
Oregon State University
Corvallis, OR 97331, USA
{mam,liang.huang}@oregonstate.edu

Bing Xiang **Bowen Zhou**
IBM Watson Group
T. J. Watson Research Center
Yorktown Heights, NY 10598, USA
{bingxia,zhou}@us.ibm.com

Abstract

Question classification is an important task with wide applications. However, traditional techniques treat questions as general sentences, ignoring the corresponding answer data. In order to consider answer information into question modeling, we first introduce novel group sparse autoencoders which refine question representation by utilizing group information in the answer set. We then propose novel group sparse CNNs which naturally learn question representation with respect to their answers by implanting group sparse autoencoders into traditional CNNs. The proposed model significantly outperform strong baselines on four datasets.

1 Introduction

Question classification has applications in many domains ranging from question answering to dialog systems, and has been increasingly popular in recent years. Several recent efforts (Kim, 2014; Kalchbrenner et al., 2014; Ma et al., 2015) treat questions as general sentences and employ Convolutional Neural Networks (CNNs) to achieve remarkably strong performance in the TREC question classification task.

We argue, however, that those general sentence modeling frameworks neglect two unique properties of question classification. First, different from the flat and coarse categories in most sentence classification tasks (i.e. sentimental classification), question classes often have a hierarchical structure such as those from the New York State DMV FAQ[1] (see Fig. 1). Another unique aspect of question classification is the well prepared answers for each question or question category. These answer sets generally cover a larger vocabulary (than the questions themselves) and provide richer information for each class. We believe there is a great potential to enhance question representation with extra information from corresponding answer sets.

1: Driver License/Permit/Non-Driver ID
a: *Apply for original* (49 questions)
b: *Renew or replace* (24 questions)
...
2: Vehicle Registrations and Insurance
a: *Buy, sell, or transfer a vehicle* (22 questions)
b: *Reg. and title requirements* (42 questions)
...
3: Driving Record / Tickets / Points
...

Figure 1: Examples from NYDMV FAQs. There are 8 top-level categories, 47 sub-categories, and 537 questions (among them 388 are *unique*; many questions fall into multiple categories).

To exploit the hierarchical and overlapping structures in question categories and extra information from answer sets, we consider dictionary learning (Candès and Wakin, 2008; Rubinstein et al., 2010) which is a common approach for representing samples from many correlated groups with external information. This learning procedure first builds a dictionary with a series of grouped bases. These bases can be initialized randomly or from external data (from the answer set in our case) and optimized during training through Sparse Group Lasso (SGL) (Simon et al., 2013).

To apply dictionary learning to CNN, we first develop a neural version of SGL, *Group Sparse Autoencoders* (GSAs), which to the best of our knowledge, is the first full neural model with group sparse constraints. The encoding matrix of GSA (like the dictionary in SGL) is grouped into different categories. The bases in different groups can be either initialized randomly or by

[1]Crawled from http://nysdmv.custhelp.com/app/home. This data and our code will be at http://github.com/cosmmb.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Short Papers), pages 335–340
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-2053

the sentences in corresponding answer categories. Each question sentence will be reconstructed by a few bases within a few groups. GSA can use either linear or nonlinear encoding or decoding while SGL is restricted to be linear. Eventually, to model questions with sparsity, we further propose novel *Group Sparse Convolutional Neural Networks* (GSCNNs) by implanting the GSA onto CNNs, essentially enforcing group sparsity between the convolutional and classification layers. This framework is a jointly trained neural model to learn question representation with group sparse constraints from both question and answer sets.

2 Group Sparse Autoencoders

2.1 Sparse Autoencoders

Autoencoder (Bengio et al., 2007) is an unsupervised neural network which learns the hidden representations from data. When the number of hidden units is large (e.g., bigger than input dimension), we can still discover the underlying structure by imposing sparsity constraints, using sparse autoencoders (SAE) (Ng, 2011):

$$J_{\text{sparse}}(\rho) = J + \alpha \sum_{j=1}^{s} KL(\rho \| \hat{\rho}_j) \tag{1}$$

where J is the autoencoder reconstruction loss, ρ is the desired sparsity level which is small, and thus $J_{\text{sparse}}(\rho)$ is the sparsity-constrained version of loss J. Here α is the weight of the sparsity penalty term defined below:

$$KL(\rho \| \hat{\rho}_j) = \rho \log \frac{\rho}{\hat{\rho}_j} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_j} \tag{2}$$

where

$$\hat{\rho}_j = \frac{1}{m} \sum_{i=1}^{m} h_j^i$$

represents the average activation of hidden unit j over m examples (SAE assumes the input features are correlated).

As described above, SAE has a similar objective to traditional sparse coding which tries to find sparse representations for input samples. Besides applying simple sparse constraints to the network, group sparse constraints is also desired when the class categories are structured and overlapped. Inspired by group sparse lasso (Yuan and Lin, 2006) and sparse group lasso (Simon et al., 2013), we propose a novel architecture below.

2.2 Group Sparse Autoencoders

Group Sparse Autoencoder (GSA), unlike SAE, categorizes the weight matrix into different groups. For a given input, GSA reconstructs the input signal with the activations from only a few groups. Similar to the average activation $\hat{\rho}_j$ for sparse autoencoders, GSA defines each grouped average activation for the hidden layer as follows:

$$\hat{\eta}_p = \frac{1}{mg} \sum_{i=1}^{m} \sum_{l=1}^{g} \|h_{p,l}^i\|_2 \tag{3}$$

where g represents the size of each group, and $\hat{\eta}_j$ first sums up all the activations within p^{th} group, then computes the average p^{th} group respond across different samples' hidden activations.

Similar to Eq. 2, we also use KL divergence to measure the difference between estimated intra-group activation and global group sparsity:

$$KL(\eta \| \hat{\eta}_p) = \eta \log \frac{\eta}{\hat{\eta}_p} + (1-\eta) \log \frac{1-\eta}{1-\hat{\eta}_p} \tag{4}$$

where G is the number of groups. Then the objective function of GSA is:

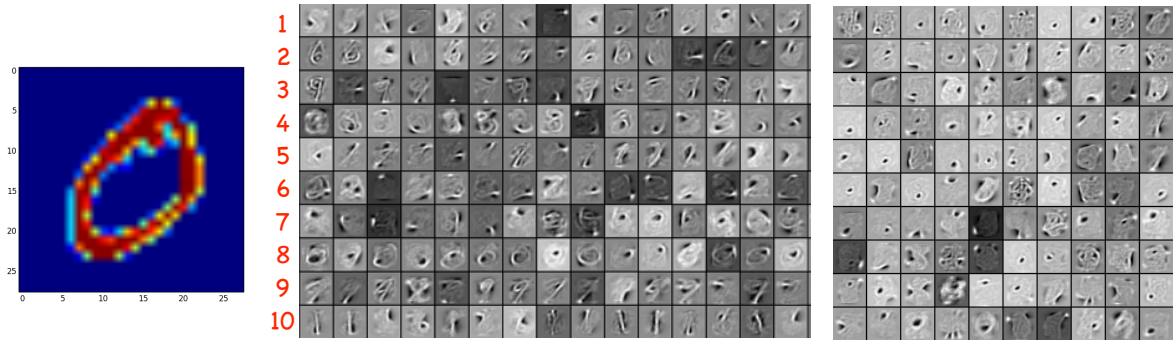
$$\begin{aligned} J_{\text{groupsparse}}(\rho, \eta) = J &+ \alpha \sum_{j=1}^{s} KL(\rho \| \hat{\rho}_j) \\ &+ \beta \sum_{p=1}^{G} KL(\eta \| \hat{\eta}_p) \end{aligned} \tag{5}$$

where ρ and η are constant scalars which are our target sparsity and group-sparsity levels, resp. When α is set to zero, GSA only considers the structure between difference groups. When β is set to zero, GSA is reduced to SAE.

2.3 Visualizing Group Sparse Autoencoders

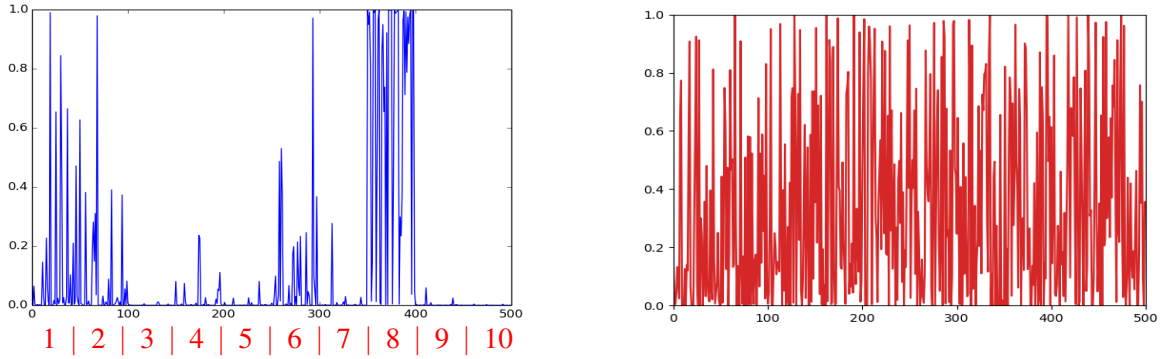
In order to have a better understanding of GSA, we use the MNIST dataset to visualize GSA's internal parameters. Fig. 2 and Fig. 3 illustrate the projection matrix and the corresponding hidden activations. We use 10,000 training samples. We set the size of the hidden layer to 500 with 10 groups. Fig. 2(a) visualizes the input image for hand written digit 0.

In Fig. 2(b), we find similar patterns within each group. For example, group 8 has different forms of digit 0, and group 9 includes different forms of digit 7. However, it is difficult to see any meaningful patterns from the projection matrix of basic autoencoders in Fig. 2(c).



(a) (b) (c)

Figure 2: The input figure with hand written digit 0 is shown in (a). Figure (b) is the visualization of trained projection matrix $\mathbf{W}$ on MNIST dataset. Different rows represent different groups of $\mathbf{W}$ in Eq. 5. For each group, we only show the first 15 (out of 50) bases. The red numbers on the left side are the indices of 10 different groups. Figure (c) is the projection matrix from basic autoencoders.



(a) (b)

Figure 3: (a): the hidden activations $\mathbf{h}$ for the input image in Fig. 2(a). The red numbers corresponds to the index in Fig. 2(b). (b): the hidden activations $\mathbf{h}$ for the same input image from basic autoencoders.

Fig. 3(a) shows the hidden activations with respect to the input image of digit 0. The patterns of the 10^{th} row in Fig. 2(b) are very similar to digit 1 which is very different from digit 0 in shape. Therefore, there is no activation in group 10 in Fig. 3(a). The majority of hidden layer activations are in groups 1, 2, 6 and 8, with group 8 being the most significant. When compared to the projection matrix visualization in Fig. 2(b), these results are reasonable since the 8^{th} row has the most similar patterns of digit 0. However, we could not find any meaningful pattern from the hidden activations of basic autoencoder as shown in Fig. 3(b).

GSA could be directly applied to small image data (e.g. MINIST dataset) for pre-training. However, in tasks which prefer dense semantic representations (e.g. sentence classification), we still need CNNs to learn the sentence representation automatically. In order to combine advantages from GSA and CNNs, we propose Group Sparse Convolutional Neural Networks below.

3 Group Sparse CNNs

CNNs were first proposed by (LeCun et al., 1995) in computer vision and adapted to NLP by (Collobert et al., 2011). Recently, many CNN-based techniques have achieved great successes in sentence modeling and classification (Kim, 2014; Kalchbrenner et al., 2014).

Following sequential CNNs, one dimensional convolutions operate the convolution kernel in sequential order $\mathbf{x}_{i,j} = \mathbf{x}_i \oplus \mathbf{x}_{i+1} \oplus \cdots \oplus \mathbf{x}_{i+j}$, where $\mathbf{x}_i \in \mathbb{R}^e$ represents the e dimensional word representation for the i-th word in the sentence, and $\oplus$ is the concatenation operator. Therefore $\mathbf{x}_{i,j}$ refers to concatenated word vector from the i-th word to the $(i+j)$-th word in sentence.

A convolution operates a filter $\mathbf{w} \in \mathbb{R}^{n \times e}$ to a window of n words $\mathbf{x}_{i,i+n}$ with bias term b' by $a_i = \sigma(\mathbf{w} \cdot \mathbf{x}_{i,i+n} + b')$ with non-linear activation

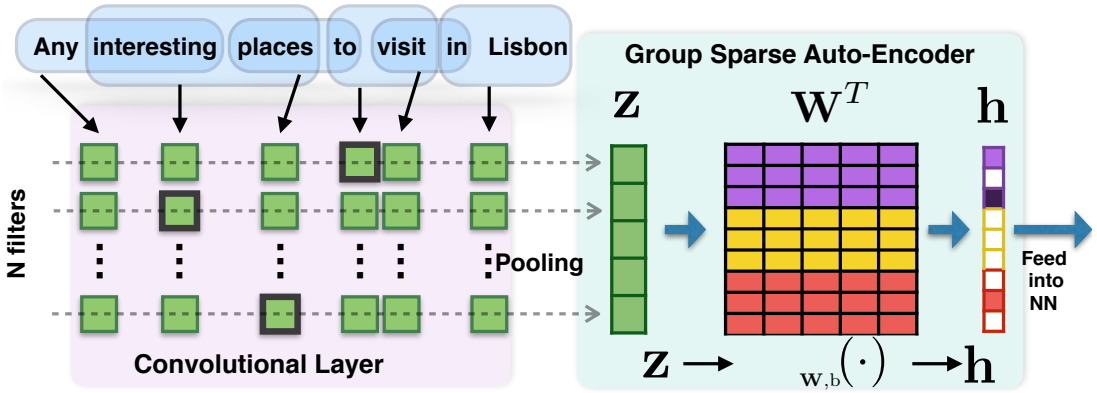


Figure 4: Group Sparse CNN. We add an extra dictionary learning layer between sentence representation $\mathbf{z}$ and the final classification layer. $\mathbf{W}$ is the projection matrix (functions as a dictionary) that converts $\mathbf{z}$ to the group sparse representation $\mathbf{h}$ (Eq. 5). Different colors in the projection matrix represent different groups. We show $\mathbf{W}^\intercal$ instead of $\mathbf{W}$ for presentation purposes. Darker colors in $\mathbf{h}$ mean larger values and white means zero.

function σ to produce a new feature. The filter $\mathbf{w}$ is applied to each word in the sentence, generating the feature map $\mathbf{a} = [a_1, a_2, \cdots, a_L]$ where L is the sentence length. We then use $\hat{a} = \max\{\mathbf{a}\}$ to represent the entire feature map after max-pooling.

In order to capture different aspects of patterns, CNNs usually randomly initialize a set of filters with different sizes and values. Each filter will generate a feature as described above. To take all the features generated by N different filters into count, we use $\mathbf{z} = [\hat{a_1}, \cdots, \hat{a_N}]$ as the final representation. In conventional CNNs, this $\mathbf{z}$ will be directly fed into classifiers after the sentence representation is obtained, e.g. fully connected neural networks (Kim, 2014). There is no easy way for CNNs to explore the possible hidden representations with underlaying structures.

In order to exploit these structures, we propose Group Sparse Convolutional Neural Networks (GSCNNs) by placing one extra layer between the convolutional and the classification layers. This extra layer mimics the functionality of GSA from Section 2. Shown in Fig. 4, after the conventional convolutional layer, we get the feature map $\mathbf{z}$ for each sentence. In stead of directly feeding it into a fully connected neural network for classification, we enforce the group sparse constraint on $\mathbf{z}$ in a way similar to the group sparse constraints on hidden layer in GSA from Sec. 2. Then, we use the sparse hidden representation $\mathbf{h}$ in Eq. 5 as the new sentence representation, which is then fed into a fully connected neural network for classification. The parameters $\mathbf{W}$ in Eq. 5 will also be fine tunned during the last step.

Different ways of initializing the projection matrix in Eq. 5 can be summarized below:

- **Random Initialization**: When there is no answer corpus available, we first randomly initialize N vectors to represent the group information from the answer set. Then we cluster these N vectors into G categories with g centroids for each category. These centroids from different categories will be the initialized bases for projection matrix $\mathbf{W}$ which will be learned during training.

- **Initialization from Questions**: Instead of using random initialized vectors, we can also use question sentences for initializing the projection matrix when the answer set is not available. We need to pre-train the sentences with CNNs to get the sentence representation. We then select G largest categories in terms of number of question sentences. Then we get g centroids from each category by k-means. We concatenate these $G \times g$ vectors to form the projection matrix.

- **Initialization from Answers**: This is the most ideal case. We follow the same procedure as above, with the only difference being using the answer sentences in place of question sentences to pre-train the CNNs.

4 Experiments

Since there is little effort to use answer sets in question classification, we did not find any suit-

Datasets	C_t	C_s	N_{data}	N_{test}	N_{ans}	Multi-label
TREC	6	50	5952	500	-	No
INSURANCE	-	319	1580	303	2176	Yes
DMV	8	47	388	50	2859	Yes
YAHOO Ans	27	678	8871	3027	10365	No

Table 1: Summary of datasets. C_t and C_s are the numbers of top-level and sub- categories, resp. N_{data}, N_{test}, N_{ans} are the sizes of data set, test set, and answer set, resp. Multilabel means each question can belong to multiple categories.

able datasets which are publicly available. We collected two datasets ourselves and also used two other well-known ones. These datasets are summarized in Table 1. INSURANCE is a private dataset we collected from a car insurance company's website. Each question is classified into 319 classes with corresponding answer data. All questions which belong to the same category share the same answers. The DMV dataset is collected from New York State the DMV's FAQ website. The YAHOO Ans dataset is only a subset of the original publicly available YAHOO Answers dataset (Fleming et al., 2012; Shah and Pomerantz, 2010). Though not very suitable for our framework, we still included the frequently used TREC dataset (factoid question type classification) for comparison.

We only compare our model's performance with CNNs for two following reasons: we consider our "group sparsity" as a modification to the general CNNs for grouped feature selection. This idea is orthogonal to any other CNN-based models and can be easily applied to them; in addition, as discussed in Sec. 1, we did not find any other model in comparison with solving question classification tasks with answer sets.

There is crucial difference between the INSURANCE and DMV datasets on one hand and the YAHOO set on the other. In INSURANCE and DMV, all questions in the same (sub)category share the same answers, whereas YAHOO provides individual answers to each question.

For multi-label classification (INSURANCE and DMV), we replace the softmax layer in CNNs with a sigmoid layer which predicts each category independently while softmax is not.

All experimental results are summarized in Table 2. The improvements are substantial for INSURANCE and DMV, but not as significant for YAHOO and TREC. One reason for this is the

	TREC	INSUR.	DMV	YAHOO dataset		
				sub	top	unseen
CNN†	93.6	51.2	60	20.8	53.9	47
+sparsity‡	93.2	51.4	62	20.2	54.2	46
$\mathbf{W}_R$	93.8	53.5	62	21.8	54.5	48
$\mathbf{W}_Q$	**94.2**	53.8	64	22.1	54.1	48
$\mathbf{W}_A$	-	**55.4**	**66**	**22.2**	**55.8**	**53**

Table 2: Experimental results. Baselines: †sequential CNNs ($\alpha = \beta = 0$ in Eq. 5), ‡CNNs with global sparsity ($\beta = 0$). $\mathbf{W}_R$: randomly initialized projection matrix. $\mathbf{W}_Q$: question-initialized projection matrix. $\mathbf{W}_A$: answer set-initialized projection matrix. There are three different classification settings for YAHOO: subcategory, top-level category, and top-level accuracies on unseen sub-labels.

questions in YAHOO/TREC are shorter, which makes the group information harder to encode. Another reason is that each question in YAHOO/TREC has a single label, and thus can not fully benefit from group sparse properties.

Besides the conventional classification tasks, we also test our proposed model on an unseen-label case. In these experiments, there are a few sub-category labels that are not included in the training data. However, we still hope that our model could still return the correct parent category for these unseen subcategories at test time. In the testing set of YAHOO dataset, we randomly add 100 questions whose subcategory labels are unseen in training set. The classification results of YAHOO-unseen in Table 2 are obtained by mapping the predicted subcategories back to top-level categories. The improvements are substantial due to the group information encoding.

5 Conclusions

In order to better represent question sentences with answer sets and group structure, we first presented a novel GSA framework, a neural version of dictionary learning. We then proposed group sparse convolutional neural networks by embedding GSA into CNNs, which result in significantly better question classification over strong baselines.

Acknowledgment

We thank the anonymous reviewers for their suggestions. This work is supported in part by NSF IIS-1656051, DARPA FA8750-13-2-0041 (DEFT), DARPA XAI, a Google Faculty Research Award, and an HP Gift.

References

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. 2007. Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems 19*.

Emmanuel J. Candès and Michael B. Wakin. 2008. An Introduction To Compressive Sampling. In *Signal Processing Magazine, IEEE*. volume 25. http://dx.doi.org/10.1109/msp.2007.914731.

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

Simon Fleming, Dan Chalmers, and Ian Wakeman. 2012. A deniable and efficient question and answer service over ad hoc social networks. In *Information Retrieval*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 1746–1751. http://www.aclweb.org/anthology/D14-1181.

Y. LeCun, L. Jackel, L. Bottou, A. Brunot, C. Cortes, J. Denker, H. Drucker, I. Guyon, U. Mller, E. Sckinger, P. Simard, and V. Vapnik. 1995. Comparison of learning algorithms for handwritten digit recognition. In *International Conference on Artificial Neural Networks*. pages 53–60.

Mingbo Ma, Liang Huang, Bing Xiang, and Bowen Zhou. 2015. Dependency-based convolutional neural networks for sentence embedding. In *Proceedings of ACL 2015*.

Andrew Ng. 2011. Sparse autoencoder. In *CS294A Lecture notes*. Stanford University, page 72.

R. Rubinstein, A. M. Bruckstein, and M. Elad. 2010. Dictionaries for sparse representation modeling. In *Neural Computation*.

Chirag Shah and Jefferey Pomerantz. 2010. Evaluating and predicting answer quality in community qa. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, USA.

Noah Simon, Jerome Friedman, Trevor Hastie, and Rob Tibshirani. 2013. A sparse-group lasso. In *Journal of Computational and Graphical Statistics*.

Ming Yuan and Yi Lin. 2006. Model selection and estimation in regression with grouped variables. In *Journal of the Royal Statistical Society*. volume 68, pages 49–67.