CSC2515: Reading Assignment
Kopal Garg

1. Jeffrey Pennington, Richard Socher, and Christopher D. Manning, “GloVe: Global Vectors for Word Representation,” Empirical Methods in Natural Language Processing (EMNLP), 2014.

Summary:

In this paper, Manning et al. (2014), introduce a new global log-bilinear regression model called GloVe that combines the benefits of two context-based embedding families: global matrix factorization methods and local context window methods. Global matrix factorization methods like Latent Semantic Analysis (LSA) decompose large matrices into low-rank approximations where rows represent words and columns represent different documents in the corpus. Another such method called Hyperspace Analogue to Language (HAL) generates matrices where each element corresponds to word occurrence frequency in the context of another word (e.g how many times apple was used in the context of juice), called co-occurrence matrices. The problem with these methods is that they perform poorly on word-analogy tasks because the similarity measure is disproportionately affected by frequently occurring words. Another set of methods called local context window methods like continuous bag-of-words and skip-gram models learn representations using adjacent words but do not utilize the global co-occurrence counts effectively as they operate locally. GloVe aims to capture the relationship between words using ratios of co-occurrence probabilities instead of looking at raw probability. Some co-occurrences that aren’t frequent or those that carry less information are dealt with using weighted least squares regression models. The proposed approach generates 2 word vectors which would differ only due to initializations if the matrix is symmetric, and builds a homomorphic function that predicts the ratios given these two word vectors and a context word vector. This is used to set up an objective function, aimed at minimizing the weighted least square problem. The weight function gives high importance to frequent co-occurrences and low importance to infrequent co-occurrences. Through experimentation, they show that this method significantly outperformed other state-of-the-art methods at that time.

Research Directions:

GloVe is currently among the most accurate and usable word embedding methods which can convert words into meaningful vectors. But this method doesn’t account for useful sentiment information of texts and also requires a large corpus of texts for training and generating exact vectors, i.e. inputs of deep learning models. This method also doesn’t consider the larger context of the text corpus. Additionally, some corpuses are smaller than what might be required to achieve high accuracy on data-driven tasks. A solution could be to use large publicly available datasets like Google News or English or Chinese Wikipedia, and then use pre-trained word embeddings to increase task accuracy.

References:

J. Pennington, R. Socher and C. Manning, "Glove: Global Vectors for Word Representation", *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014. Available: 10.3115/v1/d14-1162 [Accessed 10 December 2021].

2. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, "*Imagenet classification with deep convolutional neural networks,*" Advances in Neural Information Processing Systems (NeurIPS), 2012.

Summary:

In this paper, Krizhevsky et al (2012)., provided a proof of concept for the largest and highly optimized GPU implementation for a CNN trained on subsets (roughly 1000 images/category for 1000 categories) of ImageNet, a dataset containing millions of labeled images belonging to 20,000+ categories. Firstly, since ImageNet had images of varying resolutions, they downsampled their images to a 256x256 fixed resolution, as their system required a constant input dimensionality (as do many current out-of-the box CNN implementations). They did not perform further preprocessing except for subtracting the mean activity over training set from each pixel. The inputs to their network were raw and centered RGB pixel values. Their overall network consisted of 5 convolutional and 3 fully-connected layers. They introduce novel features to improve classification performance and decrease training time. For example, they showed that deep CNN with ReLU nonlinearity reached a 25% error rate on CIFAR-10 6x faster than the same network with traditional saturating neuron models with tanh, and also used cross-GPU parallelization to make GPUs communicate only in certain layers to improve computation. They also introduce effective techniques like enlarging the dataset using label-preserving transformations (image translations and horizontal reflections, and PCA on the RGB pixel set) and using dropout in the first two fully-connected layers (e.g. dropout of 0.5 implies the probability of each hidden neuron being set to 0 with a probability of .5) to deal with overfitting despite having 1.2 million labeled training examples. The model was trained using SGD with a minimal weight decay which, instead of being a mere regularizer, ended up assisting the model in learning. They also point out that removing even a single convolutional layer degrades their model's performance.

Research Directions:

AlexNet had 60 million parameters, a major issue in terms of overfitting. The authors do present some label-preserving transformation methods, but additional image augmentation methods could potentially help increase volume of the dataset and also help improve the performance of deep networks on the ImageNet database. For instance, including random crops, horizontal flipping and random lighting changes and other data augmentation techniques like manipulating the image brightness, and contrast could be useful. I also read a study that showed that Imagenet had "benchmark task misalignment", where a noisy data collection pipeline can lead to a discrepancy between the model's performance on a benchmark dataset and its performance on the real world database. Using the entire dataset is prohibitively expensive so the authors used an earlier downsampled version of ImageNet. However, using ImageNet as the benchmark dataset might not be ideal if benchmark task misalignment is an issue. Experiments on alternatives like CIFAR-10 (which is smaller than ImageNet but is less complex) could be dramatically faster and could help deal with this.

References:

A. Krizhevsky, I. Sutskever and G. Hinton, "ImageNet classification with deep convolutional neural networks", *Communications of the ACM*, vol. 60, no. 6, pp. 84-90, 2017. Available: 10.1145/3065386 [Accessed 10 December 2021].

3. Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, "*Deep Residual Learning for Image Recognition*," CVPR, 2016.

In this paper, He et al. (2016), introduce Residual Neural Networks, a popular CNN architecture for image classification. They empirically show that there is a max. threshold for depth when it comes to traditional CNN models, and attribute deep CNN's high performance to its number of layers. They show that as deeper networks start converging, their accuracy gets saturated, and this is followed by rapid degradation of training accuracy. However, this degradation isn't due to overfitting because if that was, it could be dealt with regularization parameters like dropout or l2-norms. Instead, this may be due to the vanishing gradient problem, an event where a network is unable to fully propagate useful gradient information from output end to input end of the model (i.e. gradient becomes too small and this causes little change in weight values). To address this problem, this paper introduces a shortcut connection across layers which helps the model learn a residual mapping equivalent to the desired hidden mapping minus the input to the residual layer, i.e. $f(x) = H(x) - x$, where $f(x)$ should be driven to 0 by the learned weights, if the identity mapping is optimal. An advantage of using this method is that shortcut connections (for identity mapping) don't require additional parameters. They perform size transformations through zero-padding (no parameters) or projections (which introduce additional parameters and perform slightly better). In comparison to VGG, ResNets are computationally cheaper but significantly deeper, and more accurate. The skip connections between layers add the outputs from previous layers to the outputs of stacked layers, enabling the training of much deeper networks than previously possible.

Research Directions:

The original ResNet had 34 layers and used 2-layer blocks. More advanced variants of ResNets that make use of more 3-layer bottleneck blocks like ResNet-50, ResNet101 or ResNet152 could be used in increasing accuracy and decreasing training time. Lots of new studies have also reported improvements due to enhanced training methods. I think it's important to separate improvements due to training methods vs architectures, as training methods like image augmentation are task-specific (and may not transfer across tasks) in comparison to architectural improvements, and may not generalize as well as architectural improvements. Architectural changes can improve performance but increase complexity and increase runtime. When designing future methods, this tradeoff should be kept in mind.

References:

K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition", *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. Available: 10.1109/cvpr.2016.90 [Accessed 15 December 2021].

4. Volodymyr Mnih, Koray Kavukcuoglu, David Silver, et al., “Human-level control through deep reinforcement learning,” Nature, 2015.

In this paper Hassabis et al. (2015), combine concepts from deep neural networks (DNN) and reinforcement learning (RL) to demonstrate a novel algorithm called deep Q-network (DQN), which they showed to outperform ML methods in 43/49 games and a diverse array of challenging tasks. The central idea behind this framework is that it prescribes how agents should act in an environment in order to maximize future cumulative reward (like the game score). Taking high dimensional raw pixel values and game scores as inputs, the general purpose agent transforms pixels into actions to be executed in real time. At its core is an optimal value function which is the max. total sum of rewards discounted by a factor $\in [0, 1]$ at each time step. Setting the discount factor to 0 results in a single-step RL problem where the reward at the current time step is important. Setting it to 1 leads to a system in which the total reward must be maximized. The total is achieved from $P(a|s)$ where a is the action and s is the observation made. Q-learning updates on samples drawn from a collection of stored samples are used at each iteration. It overcomes instability in estimating Q through incorporating ‘replay of experiences. They also employ data processing techniques to simplify (remove flickering, grayscale conversion, downsampling, etc.) the original input which is pixel values from the Atari emulator, and the score. Ultimately, they demonstrate an iterative reward system that can take in and learn from environmental inputs, perform some actions, and assess the consequence of its actions by relating the actions to rewards over time (Q, which is estimated by DNN). Their experimental results show that the agent scored >=75% of the human score on >50% of the games, and outperformed current ML methods in 43/39 challenging tasks. Experience replay and a separate network for evaluating targets are two major advancements presented in this paper.

Research Directions:

As pointed out by several subsequent studies, Q-iteration makes an assumption that the state of the environment is fully observable. This may not translate so well in real-world tasks. The method chooses an optimum action from a state input and is evaluated in a simulated environment, where it is easy for the simulator to obtain input, to perform actions, and to get a reward. In real-world cases, the input from a sensor may contain complex background noises, and the input states may be more diverse. Consequently, more training may be required. As pointed out by Kimura et al. (2018), generative models (with a random policy agent or inputs without actions) may be used to initialize the network for reducing the number of training trials.

Additionally, the method performed poorly on a few tasks like adventure games, as they require semantic understanding, so ways of incorporating this information are certainly worth investigating.

Finally, future methods can work on improving the experience replay algorithm.

References:

V. Mnih et al., "Human-level control through deep reinforcement learning", *Nature*, vol. 518, no. 7540, pp. 529-533, 2015. Available: 10.1038/nature14236 [Accessed 15 December 2021].

D. Kimura, "DAQN: Deep Auto-encoder and Q-Network", 2018. Available: http://arXiv:1806.00630v1. [Accessed 15 December 2021].

5. Ronan Collobert and Jason Weston, "*A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning*," International Conference on Machine Learning (ICML), 2008.

Summary:

Note: I apologize that the summary for this particular paper went over the limit. I really enjoyed reading it, and wanted to understand the architecture in greater detail. **Blue text is optional architecture-related details.**

Using a unified convolutional neural network architecture trained jointly using weight-sharing on several Natural Language Processing (NLP) tasks, Weston et al. (2008), show that both multi-task learning (MTL) and semi-supervised learning improve generalization of shared tasks. Common failings of current approaches are that they feed shallow, linear classifiers with hand-engineered features like sentences transformed into vectors. This is a largely empirical and trial-and-error based approach, and is, therefore, difficult to scale. Features learned in separate tasks are cascaded and this propagates error. Given an input sentence, the proposed CNN architecture learns features relevant to tasks at hand (with limited prior knowledge) and outputs a host of semantic and syntactic subtask predictions like part-of-speech tags, named entity tags, semantic role-labeling (SRL), likelihood that a sentence makes grammatical and semantic sense, etc. The language model is trained in an semi-supervised manner on the entire Wikipedia corpus. Rest of the tasks are supervised and require labeled training data.

In terms of the architecture, it first converts raw words into real-valued vectors for subsequent processing. Using a lookup table, each word is embedded into a d-dimensional space: $LT_W(i) = W_i$, where $W \in R^{dx|D|}$ represents parameters to be learnt (through backpropagation), and $W_i \in R^d$ represents the ith column of W and d is the word vector size. Basic pre-processing steps like stemming, lemmatization and lower-casing can also be employed. After decomposing words into features, and associating each element with a lookup table, each word is embedded into a d-dimensional space, by concatenating lookup table outputs to get, $LT_{W^1,\ldots,W^K}(i)^T$. Next, an additional feature that encodes relative distance with respect to the chosen predicate is added, as the class label for each word depends on the given predicate. One disadvantage of using normal NNs is that they can't handle sequences of varying lengths. The method therefore employs a window-approach which uses a fixed window around the word to be labeled. However, this approach fails on complex tasks like SRL where the label for a word may depend on some distant word outside the fixed window in consideration. This method proposes the use of Time-Delay NN (TDNN) to model such long distance dependencies more efficiently. TDNN reads sequences in a sequential and online fashion and performs a convolution. It considers all windows of a fixed size in a sentence. The TDNN is then fed into an additional "Max" layer that aims to capture the most relevant local features produced by the convolutional layer. Using this approach of extracting features from all windows of fixed size, the method computes the label for the word of interest. For simpler tasks, TDNN performs a linear operator over the input sequence, but for more complicated tasks, non-linear models are required. The output layer is computed through backpropagation, and its size depends on #classes considered in the task. This is then followed by a softmax layer which ensures output probabilities sum to 1, s.t. $p_i = e^{o_i^{last}} / \Sigma_j e^{o_i^{last}}$.

Multi-task learning involves utilizing features from one task for another auxiliary task. Since in this method the deepest (lookup tables) layer implicitly learns relevant features, we can assume that when the TDNN is trained on related tasks, sharing these layers would improve features, and lead to better generalization performance. This means that the tasks can share a look-up table, but the final layer of the

network can be task-specific. Next in this approach they propose to train the model on supervised NLP tasks on labeled data and unsupervised tasks on unlabeled data. The language model performs a binary classification task, deciphering whether the word in a window is context-specific or not, and this is done using all possible windows from the entire Wikipedia dictionary. Finally, they conduct some experimentation to confirm the feasibility of using deep NN architectures for NLP tasks without explicit syntactic features. Therefore, it isn't always necessary to use syntax as features for semantic extraction in NLP tasks.

Research Directions:

Weston et al. (2008), showed that word embeddings can be used in place of common features on many NLP tasks. A next step could involve a systematic comparison between word representations where we can investigate whether model performance could benefit from using other unsupervised word embeddings like Brown cluster embeddings, Hierarchical log-bilinear model embeddings etc (as shown by Bengio et al. (2010). It might also be worth investigating whether combining various word embeddings could lead to better generalization performance.

References:

Turian, J., Ratinov, L. and Bengio, Y., 2010. Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, [online] pp.384–394. Available at: <https://aclanthology.org/P10-1040> [Accessed 6 December 2021].

R. Collobert and J. Weston, "A unified architecture for natural language processing", *Proceedings of the 25th international conference on Machine learning - ICML '08*, 2008. Available: 10.1145/1390156.1390177 [Accessed 9 December 2021].