10-707: Advanced Deep Learning **Spring 2020**

Lecture 19: April 8

Lecturer: Andrej Risteski *Scribes: Wael Al Saeed*

Note: *LaTeX template courtesy of UC Berkeley EECS dept.*

Disclaimer: *These notes have not been subjected to the usual scrutiny reserved for formal publications, if as reader, you find an issue you are encouraged to clarify it on Piazza. They may be distributed outside this class only with the permission of the Instructor.*

19.1 Review: Unsupervised Learning

The Unsupervised Learning problem of learning from the data with the absence of labels can be formulated as solving one of the three following interrelated tasks:

1. **Structure Learning:** Fitting a parametrized structure to the data to reveal information (e.g. clustering).
2. **Distribution Learning:** Learning a (parametrized) distribution close to the data generating distribution.
3. **Representation/Feature Learning:** Learning a (parametrized) distribution that implicitly reveals an embedding/representation of the data for downstream tasks.

19.2 Self-supervised/Predictive Learning

Self-supervised/Predictive Learning is a Representation Learning method that aims to design Supervised Learning tasks for an unlabeled data that induces a good representation for downstream tasks. The concept doesn't have a well formalized mathematical definition, but the intuition is that it aims to design Supervised Learning tasks that are interesting and difficult enough that any predictor that does well on these tasks is expected to have learned semantically-meaningful information about the data.

An common way to create such tasks is predicting any part of the input from some other part, here are few examples:

- In the context of Time-Series data this could be predicting the future from the past, or the past from the present.
- In the context of image data this could be predicting the top part of an image from the bottom part of the image.

19.3 Predictive Learning in Natural Language Processing

19.3.1 Word Embedding

A Word Embedding is a semantically meaningful vector representation of words. This is not a well defined goal but here are some desirable properties of word embeddings:

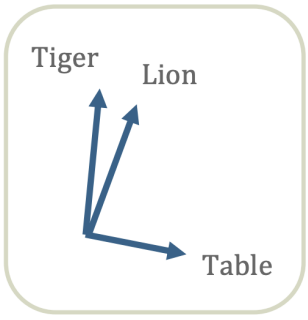

Figure 19.1: An example of word embeddings.

1. Geometric correlations of word representations such as inner product gives an indication of similarity in meaning of the words. An illustration of this idea can be seen in Figure 19.1 where the word embedding of *Lion* has a smaller angle with the word embedding of *Tiger* than the word embedding of *Table*.

2. Usefulness for some downstream tasks. One example would be effective sentiment classification by training a simple (e.g. linear) classifier. Another example is that a simple neural network can be trained on the word embeddings in two different languages and effectively translate between the two languages.

19.3.2 Word Embeddings via Predictive Learning

Here are examples of Supervised Learning tasks that can be constructed to derive word embeddings:

- **Predicting the Next Word Given a Few Previous Words**

  This is one of the most basic tasks that could be constructed, where the target is the next word x_t given the previous L words $x_{t-L}, \ldots, x_{t-1}$ as input. In this case the task can be formalized as:

$$\max_{\theta} \sum_{t} \log\ p_\theta(x_t | x_{t-1}, \ldots, x_{t-L})$$

Figure 19.2: An illustration of predicting the next word given a number of previous words as input.

  This task can be motivated by the common assumption in NLP that the underlying distribution is Markov, that is the distribution of a word is fully determined by the values of the previous L words $p(x_t|x_{t-1}, \ldots, x_{t-L}) = p(x_t|x_{t-1}, \ldots, x_1)$.

  This formalization suffers from the following two problems:

  1. The Markov assumption will not hold if the model aims to capture long texts such as books.
  2. Parameterizing the model $p_\theta(x_t|x_{t-1}, \ldots, x_{t-L})$, at least in the trivial way, would require $\mathcal{O}(|V|^L)$ parameters which doesn't scale well.

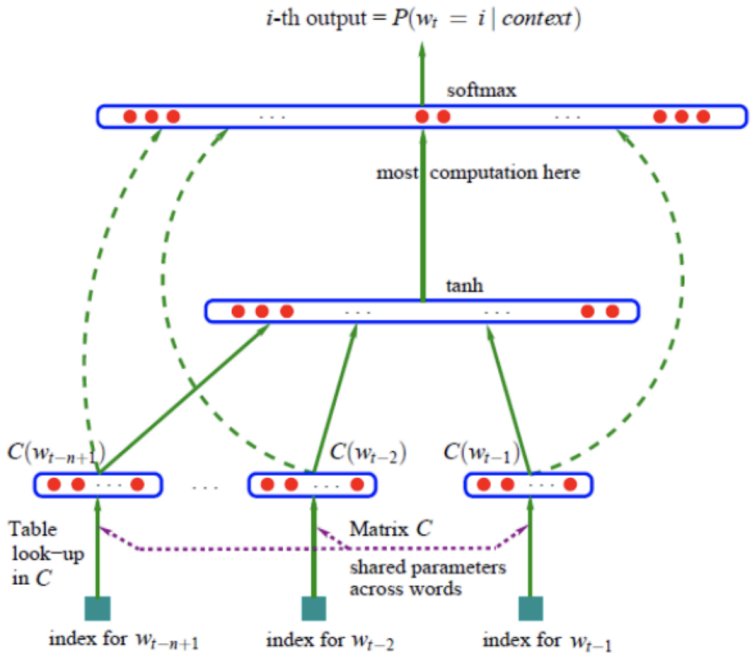

Figure 19.3: An illustration of the Neural Parameterization proposed in [BDV03].

On of the first famous papers to utilize Deep Learning to address the second problem was [BDV03] which introduced many of the concepts used to derive word embeddings via Predictive Learning. Some of the main ingredients of the model are:

- Word Embeddings $C(w)$ for all words w in the dictionary.
- Non-Linear Transform, could Deep Neural Networks or Recurrent Network, which takes $C(w_{t-1})$, $\ldots$, $C(w_{t-L})$ as input and produce a vector o of the size of the vocabulary as output.
- Softmax parameterized by o to produce a probability distribution for x_t.

- **Predicting the Middle Word Given Surrounding Words**

  In this task the objective is to predict a word x_t given a the words that surrounds it in an L-width window, that is $x_{t-L}, \ldots, x_{t-1}, x_{t+1}, \ldots, x_{t+L}$, which is sometimes termed as the context of x_t. In this case the task can be formalized as:

$$\max_{\theta} \sum_{t} \log\ p_\theta(x_t | x_{t-L}, \ldots, x_{t-1}, x_{t+1}, \ldots, x_{t+L})$$

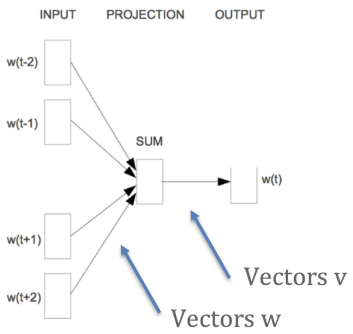

Figure 19.4: An illustration of the CBOW model [MCC13].

One of the ways to parameterize p_θ is the Continuous Bag of Words (CBOW) proposed in [MCC13]. In CBOW each word x in the vocabulary has two encodings v_x and w_x. Then the model is parametrized as:

$$p_\theta(x_t|x_{t-L},\dots,x_{t-1},x_{t+1},\dots,x_{t+L}) \propto \exp\left(\langle v_{x_t}, \sum_{\substack{t-L\le i\le t+L\\ i\neq t}} w_{x_i}\rangle\right)$$

- **Predicting Surrounding Words Given the Middle Word**

  In this task the objective is to use a word x_t as input and predict the context of the word, that is $x_{t-L},\dots,x_{t-1},x_{t+1},\dots,x_{t+L}$. In this case the task can be formalized as:

$$\max_\theta \sum_t \log\ p_\theta(x_{t-L},\dots,x_{t-1},x_{t+1},\dots,x_{t+L}|x_t)$$

  Moreover if we make the assumption that $x_{t-L},\dots,x_{t-1},x_{t+1},\dots,x_{t+L}$ are conditionally independent given x_t, then the optimization problem can be rewritten as:

$$\max_\theta \sum_t \sum_{\substack{t-L\le i\le t+L\\ i\neq t}} \log\ p_\theta(x_i|x_t)$$

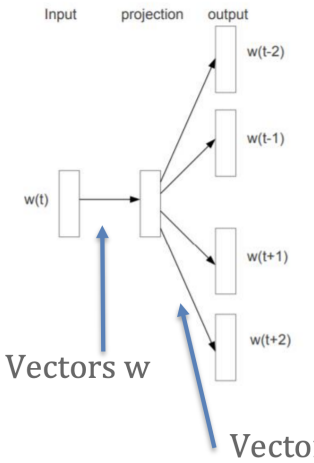

Figure 19.5: An illustration of the Skip-Gram model [MSC13].

One of the ways to parameterize p_θ is Skip-Gram proposed in [MSC13], where each word x will similarly have two encodings v_x and w_x. The density function is then parametrized as:

$$p_\theta(x_i|x_t) \propto \exp\left(\langle v_{x_i}, w_{x_t}\rangle\right)$$

One of the major issues of this model is that computing the partition function needed to normalize the distribution would require summing over the entire vocabulary, which is computationally expensive. Various tricks have been proposed to deal with this problem such as negative sampling, hierarchical softmax, etc.

- **Predicting Random 15% of the Words Given the Rest**

  In this task the objective is to predict masked 15% of the words given the rest of the words in the document. Parameterizations of this task are more complicated than the previous ones, which will not be explored in this lecture. BERT [DCL18] which is trained on this task achieves word embeddings that are currently considered the state-of-the-art for many of the downstream tasks.

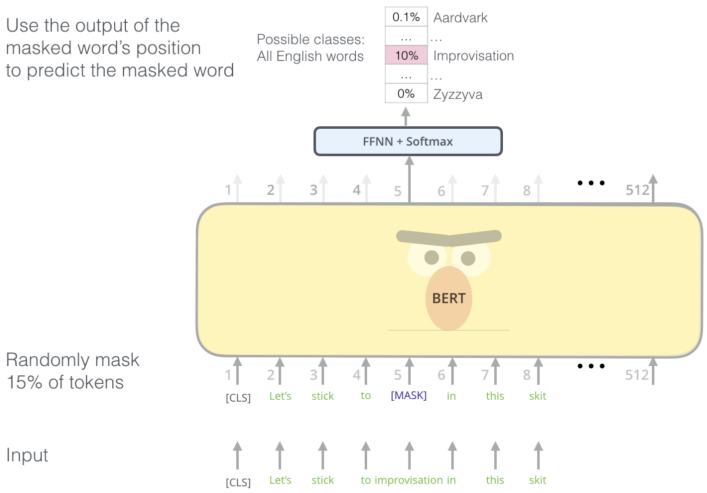

Figure 19.6: An illustration of the BERT model [DCL18].

19.3.3 Evaluating Word Embeddings

19.3.3.1 Direct Methods

Models of the first task (predicting next word given previous words) can be used as generative models for text (also called language model). In this case a natural measure to use is Cross-Entropy:

$$-\mathbb{E}_{x_1,\ldots,x_T}\left[\log\ p_\theta(x_{\leq T})\right] = \mathbb{E}_{x_1,\ldots,x_T}\left[\sum_t \log\ p_\theta(x_t|x_{<t})\right]$$

Sometimes the exponential of Cross-Entropy is used, which is termed perplexity.

19.3.3.2 Indirect Methods

Most of the tasks discussed in the previous section don't give rise to a generative model, which means that Cross-Entropy cannot be evaluated. For this reason we resort to test performances on downstream tasks as an indirect evaluation measure of word embeddings. These tasks can be divided into two categories:

Extrinsic Tasks:

The idea is to measure performance of word embedding in tasks that would benefit from a good word embedding. The main question is how well can (Supervised) downstream task perform by finetuning the embedding. Finetuning is usually understood to be training a simple model (small feedforward network) on top of the embedding. Examples of such tasks are Part-of-Speech Tagging, Named Entity Recognition, among others.

Intrinsic Tasks:

The idea is to evaluate word embeddings on tasks measuring their "Semantic" properties. Examples are:

Figure 19.7: Nearest neighbors query to "Frog" in some Word Embedding.

1. Semantic Similarity:
   This is based on the observation that the embeddings of semantically similar words tend to have larger cosine similarity (normalized inner product) than semantically unrelated words. More precisely for two words i, j with embeddings w_i, w_j the cosine similarity $\cos(w_i, w_j) = \langle \frac{w_i}{||w_i||}, \frac{w_j}{||w_j||} \rangle$ tends to be larger for semantically similar words. So to solve a query of the form "what is the most similar word to a given word", the word with the highest cosine similarity to it is chosen as output.

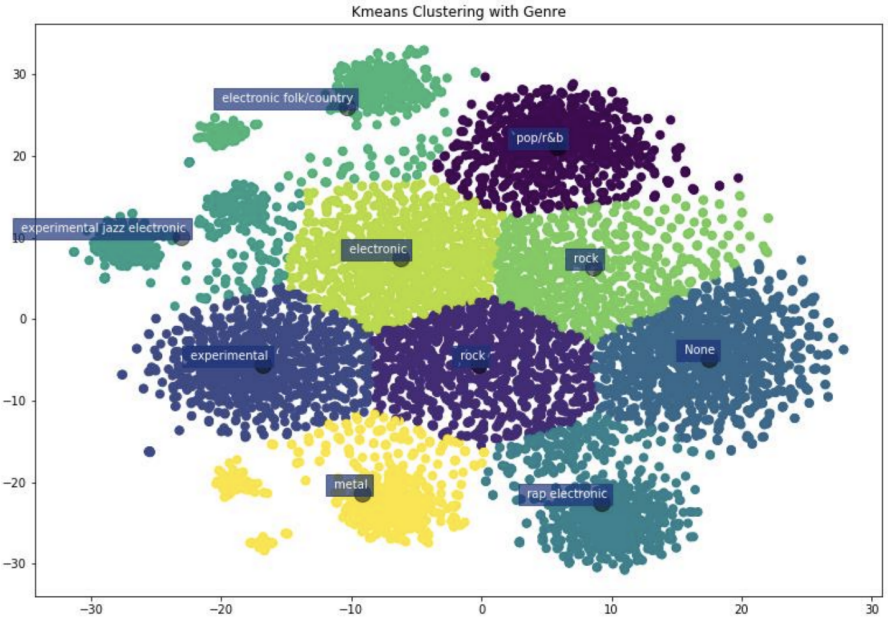

Figure 19.8: t-SNE projection of word embeddings for artists (clustered by genre). Image Source

2. Semantic Clustering:
   Clustering word embeddings should give rise to "semantically" relevant clusters.

3. Analogies:
   This is based on the observation that Analogy queries can be solved by Linear Algebra on word embeddings. As an example solving for w that would make (King, w) analogical to (Man, Woman) can be obtained by $\text{argmin}_w ||(v_w - v_{\text{King}}) - (v_{\text{Woman}} - v_{\text{Man}})||^2$, which in a good word embedding should return the word Queen.

19.4 Predictive Learning in Computer Vision

19.4.1 Inpainting Task

Inpainting is analogous to the tasks used to derive Word Embeddings, where the basic idea is to predict part of the image from the rest of the image. The intuition for why this task would result in a good embedding is that a model that performs well on completing an image is expected to have developed some understanding of the organization of an image, parts of the picture, etc.

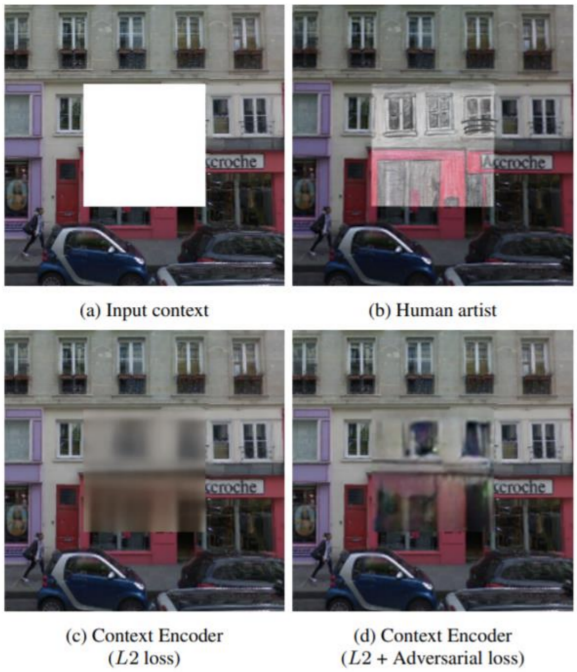

Figure 19.9: An example of two models outputs in an Inpainting task compared to a human artist.

One of the earliest papers to apply self-supervised methods in the context of images was [PKD16]. The architecture proposed in the paper is on a high level composed of an Encode and a Decoder, where the encoder derives an embedding of the image then the decoder reconstructs an image from the embedding.

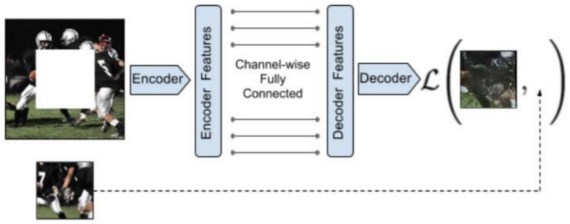

Figure 2: Context Encoder. The context image is passed through the encoder to obtain features which are connected to the decoder using channel-wise fully-connected layer as described in Section 3.1. The decoder then produces the missing regions in the image.

Figure 19.10: An overview of the architecture proposed in [PKD16].

Potential issues in this task:

1. Losses for vision tasks are more difficult to design than in NLP. When the model is trained with L_2 reconstruction loss, the reconstructed image tends to be blurry as can be seen in Figure 19.9.

2. The choice of the masked out region has a significant impact.

[PKD16] addressed the first issue by using a combination of L_2 reconstruction loss and adversarial loss. Recall that one of the usefulness of GANs was to provide a more natural loss for images. The loss that was used to train the model was the following:

$$\mathcal{L}_{\text{rec}}(x) = \left|\left|\hat{M} \odot \left(x - F((1-\hat{M}) \odot x) \odot x\right)\right|\right|_2^2$$

$$\mathcal{L}_{\text{adv}}(x) = \max_D \left(\log\left(D(x)\right) + \log\left(1 - D\left(F((1-\hat{M}) \odot x)\right)\right)\right)$$

$$\mathcal{L}(x) = \lambda_{\text{rec}}\mathcal{L}_{\text{rec}}(x) + \lambda_{\text{adv}}\mathcal{L}_{\text{adv}}(x)$$

Where $\hat{M}$ is the mask, F is the composition of the encoder and the decoder, and D is the discriminator.

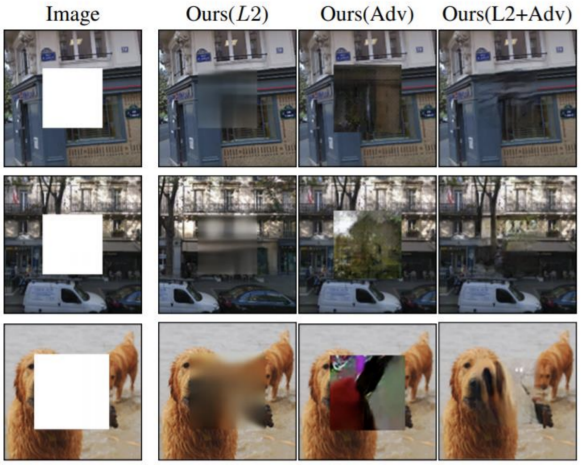

Figure 19.11: Samples of Inpainting results generated by training the model on different losses [PKD16].

The paper also addressed the second issue of region selection by examining three options:

1. Fixed (central) Region: Tends to produce less generalizable representations.

2. Random Blocks: Produces better representation than Fixed variant.

3. Random silhouette: Produces better representation than both other variants.

An intuitive conclusion of this result is that region selection should be done so that the learning task is solvable but not too easy.

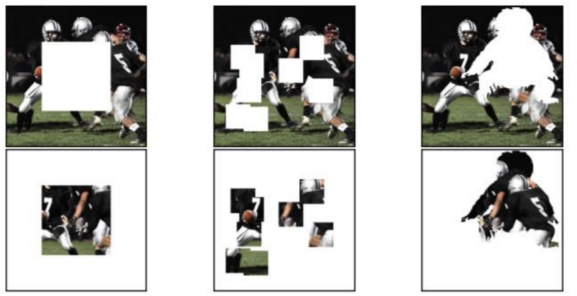

Figure 19.12: The three different region selection methods examined in [PKD16].

19.4.2 Jigsaw Puzzles Task

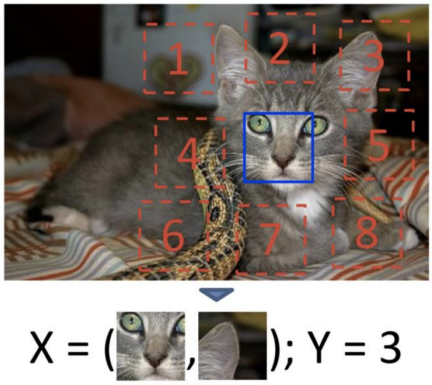

Figure 19.13: An example of a Jigsaw Puzzle.

First proposed in [DGE15], the idea is to extract pairs of square pieces of the image dataset with the task of predicting the relative ordering of the extracted pairs. The representation of the image is then taken to be the value of penultimate layer of the neural network trained to solve this task, when the image is fed to it as an input. The intuition is that understanding relative positioning of image pieces requires an understanding of how images are composed.

On the first look this idea of generating representations doesn't seem to be satisfying, as the task should be posed in such a way that the predictor cannot take "obvious" shortcuts to derive the output. Here are some of the shortcuts that a model could take to derive the output, and the way in which the authors counteracted each point:

1. Continuity in the Texture of the Boundaries. The authors addressed this issue by including gaps between tiles.

2. Long Lines that Spans the Tiles are a Clue. The authors added a jitter to the locations of the tiles so they are not perfectly aligned.

3. Chromatic aberration (some cameras tend to focus different wavelengths at different position – e.g. green shifts towards center of image). The authors randomly dropped 2 out of the 3 RGB channels.

19.4.3 Rotation Prediction Task

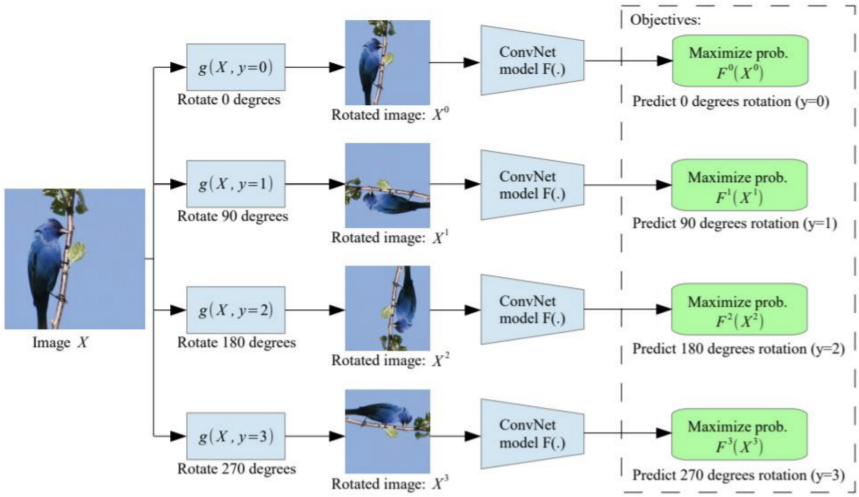

Figure 2: Illustration of the self-supervised task that we propose for semantic feature learning. Given four possible geometric transformations, the 0, 90, 180, and 270 degrees rotations, we train a ConvNet model $F(.)$ to recognize the rotation that is applied to the image that it gets as input. $F^y(X^{y^*})$ is the probability of rotation transformation y predicted by model $F(.)$ when it gets as input an image that has been transformed by the rotation transformation y^*.

Figure 19.14: A description of the Rotation Prediction Task as found in [GSK18].

Rotation Prediction task was introduced in [GSK18], where the task is to predict the rotation applied to the input image. The representation of an image is then taken to be the value of the penultimate layer of the trained neural network when applied to the image as input. The intuition of this task is that the model is trying to predict a global transformation which requires understanding of the image. Additionally given that ConvNets are better at capturing local transformations, there is no obvious way that the model could take a shortcut to derive the output.

19.4.4 Contrastive Divergence Tasks

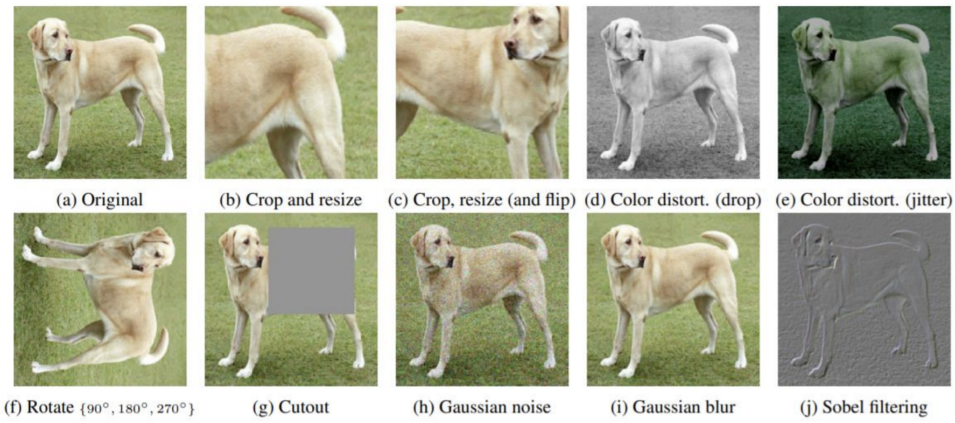

Figure 19.15: Examples of distortions.

This framework was popularized by the paper [OLV18]. The intuition behind this task is that in a "semantically" meaningful embedding, distorting an image should produce a similar representation. The task proceeds by producing multiple augmented samples of every image in the training set via applying various transformations. Then an encoder E is trained to predict whether two samples are augmentations of the same image. A common way to achieve that is to train E so that $\langle E(x), E(x')\rangle$ is big if x, x' are augmentations of the same sample, and small otherwise. An example of a loss that aims to capture this is the following:

$$l(x, x') = -\log\left(\frac{\exp(\tau\langle E(x), E(x')\rangle)}{\sum_{x,x'}\exp(\tau\langle E(x), E(x')\rangle)}\right)$$

$$\min \sum_{\substack{x,x' \text{ are augmentations} \\ \text{of the same sample}}} l(x, x')$$

The current state of the art of representations derived by self-supervising tasks on many downstream tasks is based on this framework, presented in [CKN20]. The authors of the paper observed that the augmentations that seemed to work best were compositions of a geometric augmentation (e.g. crop/rotation) and an appearance augmentation (e.g. color distortion/blur).

19.4.5 Troubling Fact: Architecture of the Classifier Matters

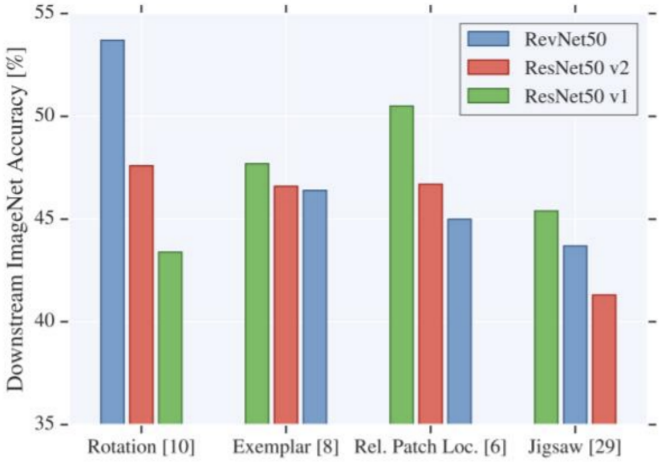

Figure 1. Quality of visual representations learned by various self-supervised learning techniques significantly depends on the convolutional neural network architecture that was used for solving the self-supervised learning task. In our paper we provide a large scale in-depth study in support of this observation and discuss its implications for evaluation of self-supervised models.

Figure 19.16: Performance of various architecture and method combinations in downstream tasks [KZB19].

While in supervised learning tasks it is usually the case that similar architectures perform similarly on the same task, [KZB19] compared the performances in downstream tasks of various self-supervised learning methods on reasonably related architectures and found significant variance in performance across architectures. This suggests that the performance of each of self-supervised learning methods of deriving image embedding is highly dependent on the architecture of the network trained.

References

[BDV03] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. "A neural probabilistic language model". J. Mach. Learn. Res. 3, null (March 2003), 1137–1155.

[MCC13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. "Efficient Estimation of Word Representations in Vector Space". In ICLR Workshop Papers.

[MSC13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. "Distributed representations of words and phrases and their compositionality". In NIPS, pages 3111–3119.

[DGE15] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. "Unsupervised visual representation learning by context prediction". In ICCV, 2015.

[PKD16] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. "Context encoders: Feature learning by inpainting". CVPR, 2016.

[DCL18] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. "Bert: Pretraining of deep bidirectional transformers for language understanding". arXiv preprint arXiv:1810.04805, 2018.

[GSK18] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. "Unsupervised representation learning by predicting image rotations". arXiv preprint arXiv:1803.07728, 2018.

[OLV18] van den Oord, A., Y. Li & O. Vinyals. 2018. "Representation learning with contrastive predictive coding". https://arxiv.org/abs/1807.03748.

[KZB19] Alexander Kolesnikov, Xiaohua Zhai, and Lucas Beyer. "Revisiting self-supervised visual representation learning". arXiv preprint arXiv:1901.09005, 2019.

[CKN20] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. "A simple framework for contrastive learning of visual representations". arXiv:2002.05709, 2020.