[AI502] 2nd Survey Paper

Victor Cappa*
KAIST School of Computing
victor.cappa@kaist.ac.kr

Abstract

This survey paper subsumes and analyzes different papers regarding deep learning related topics such as a novel neural network model called RNN Encoder-Decoder [1] consisting of two Recurrent Neural Networks (RNNs), able to successfully learn phrase representations for Statistical Machine Translation; a novel approach for training deterministic Auto-Encoders by adding a well chosen penalty term to the classical reconstruction cost function [2] allowing us to achieve superior results compared to vanilla Variational Auto-Encoders; we introduce improvements and explore the use of Vector Quantized Variational AutoEncoder (VQ-VAE) models for large scale image generation [3]; and finally, we define a form of GAN (Generative Adversarial Network) called Wasserstein-GAN [4] and we provide empirical evidence of its improvements compared to vanilla GAN both in inference and in training performances.

1 RNN Encoder-Decoder Architecture

In this section we introduce a novel deep learning architecture called RNN Encoder-Decoder Architecture [1] specifically designed to perform Statistical Machine translation, consisting in performing language statistical translation of phrases from one language to another. This is composed of an Encoder, consisting in a RNN, that permits to generate a fixed-size representation of the input variable-size phrase, and a Decoder, also consisting in a RNN that starting from the Encoder's result generates the variable-size output sequence. This RNN Encoder-Decoder Architecture learns a continuous space representation of a phrase that is proven to preserves both the semantic and syntactic structure of it.

1.1 RNN Encoder-Decoder Hidden Unit and Model Training

RNN Encoder-Decoder Hidden Unit We propose a novel hidden unit motivated by the LSTM unit but much simpler to compute and implement. It is composed of an *update* gate and a *reset gate*, where the former selects whether the hidden state is to be updated with a new hidden state h~ and the latter decides whether the previous hidden state is ignored. The actual activation of the proposed unit h_j is computed by $h_j^{\langle t \rangle} = z_j h_j^{\langle t-1 \rangle} + (1 - z_j)\tilde{h}_j^{\langle t \rangle}$ where $[.]_j$ denotes the j-th element of a vector and $\tilde{h}_j^{\langle t \rangle} = \phi\left([Wx]_j + \left[U\left(r \odot h_{\langle t-1 \rangle}\right)\right]_j\right)$. The *reset* gate r_j is computed by $r_j = \sigma\left([W_r x] + \left[U_r h_{\langle t-1 \rangle}\right]_j\right)$ where σ is the logistic sigmoid function, and x and h_{t-1} are the input and the previous hidden state. The *update* gate z_j is computed by $z_j = \sigma\left([W_z x]_j + \left[U_z h_{\langle t-1 \rangle}\right]_j\right)$. W_r, U_r, W_z and U_z are learned weight matrices.

*Master's program student at KAIST School of Computing. Daejeon, South Korea. ID number: 20206080

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

Model Training in SMT (Statistical Machine Translation) The goal Statistical Machine Translation is to find a translation vf given a source sentence ve, which maximizes $p(vf \mid ve) \propto p(ve \mid vf)p(vf)$ where the first term at the right hand side is called *translation model* and the latter *language model*. In practice, however, most SMT systems model $\log p(vf \mid ve)$ as a log-linear model with additional features and corresponding weights: $\log p(vf \mid ve) = \sum_{n=1}^{N} w_n f_n(vf, ve) + \log Z(ve)$ where f_n and w_n are the n-th feature and weight, respectively. $Z(ve)$ is a normalization constant that does not depend on the weights. The weights are optimized to maximize the BLEU score on a development set. While training we want to ensure that the model is focused toward learning linguistic regularities, i.e., distinguishing between plausible and implausible translations, in other words learning the "manifold" of plausible translations. Moreover, when we train the RNN Encoder–Decoder, we ignore the (normalized) frequencies of each phrase pair in the original corpora by means of the regularization term covered in the previously described formula.

1.2 Experimental Results

RNN Encoder–Decoder and SMT The RNN Encoder-Decoder Architecture is evaluated on the English/French translation task of the WMT'14, with a limited dataset containing 15,000 words for both English and French (covering approximately 93% of the vocabulary). The used RNN Encoder–Decoder had 1000 hidden units consisting in the discussed module and trained with Adadelta and stochastic gradient descent, with each input word was projected into the embedding space R^{512}. After training, the language model achieved a perplexity of 45.80 and BLEU score of 31.50 on the development set and 34.64 on the test set.

RNN Encoder–Decoder and Word Embedding The RNN Encoder–Decoder is a continuous space language model using neural networks, thus it is proven to be able to learn semantically meaningful embeddings capturing both semantic and syntactic structures of the phrases.

2 Contractive Auto-Encoders

In this section we propose a novel approach for training deterministic Auto-Encoders by adding a penalty term, corresponding to the Frobenius norm of the Jacobian matrix of the encoder activations with respect to the input, to the classical reconstruction cost function. This term results in a localized space contraction, thus the name *Contractive Auto-Encoder*, which in turn yields robust features on the activation layer, and helps to carve a representation that better captures the local directions of variation dictated by the data, increasing robustness to small perturbations around the training points. This is particularly important because auto-encoders are often employed as feature extractors, where the number of extracted features (consisting in the latent or hidden space) is lower or much lower than the input dimension, or to extract a higher number of features than the input dimension, yielding to a richer higher-dimensional representation. In this setup, using regularization becomes even more essential compared to the former scenario to prevent Specifically, when the encoding activation functions are linear and the number of hidden units is inferior to the input dimension (bottleneck), the learnt parameters of the encoder are a subspace of the principal components *PCA* of the input space [5]. However, non-linear activation functions are proven to be able to learn more useful feature-detectors. We have different *Auto-Encoders* variants such as *Basic Auto-Encoders (AE)*, *Regularized Auto-Encoders (AE+wd)* where we introduce regularization techniques such as weight-decay, and *Denoising Auto-Encoders (DAE)* where we corrupt the input data with additive isotropic Gaussian noise before sending it through the autoencoder.

2.1 Contractive Auto-Encoders

The Contractive auto-encoder (CAE) is obtained with a well known regularization term yielding to objective function $J_{CAE}(\theta) = \sum_{x \epsilon D_n} (L(x, g(f(x))) + \lambda \left\| J_f(x) \right\|_F^2)$, where decoder function g maps the hidden representation $h(x)$ back to a output reconstruction y, encoder function f maps input x to hidden representation h(x), λ hyper-parameter controls the strength of the regularization, and $\left\| J_f(x) \right\|_F^2$ is the Frobenius norm of the Jacobian $J_f(x)$. This final term is, in practice, the sum of squares of all partial derivatives of the extracted features with respect to input dimensions, namely

$\|J_f(x)\|_F^2 = \sum_{ij}(\frac{\partial h_j(x)}{\partial x_i})^2$ where this penalization term ensures encourages the mapping to the feature space to be contractive in the neighborhood of the training data.

3 Quantized Variational AutoEncoder (VQ-VAE)

In this section we explore the use of Vector Quantized Variational AutoEncoder (VQ-VAE) models for large scale image generation [3]. We keep simple the original VQ-VAE feed-forward encoder and decoder networks, while using hierarchical multi-scale latent maps for increased resolution, as well as autoregressive priors such as PixelCNN.

Generative models overview There are two main types of generative models: likelihood based models, such as VAEs [6], flow based models [7] and autoregressive models [8], and implicit generative models such as Generative Adversarial Networks (GANs) [11]. Likelihood based methods optimize a negative log-likelihood (NLL) loss function on top of the training data. This objective allows model-comparison and measuring generalization to unseen data, and do not suffer from the problems of mode collapse and lack of diversity that can be find in GAN models. These phenomenons happen because of the NLL objective function, since the probability that the model assigns to all examples in the training set is maximized.

Vanilla VQ-VAE Vanilla VQ-VAE consists in an encoder that maps observations onto a sequence of discrete latent variables, and a decoder that reconstructs the observations from these discrete variables, both sharing the same codebook. The encoder performs a non-linear mapping from the input space x, to a vector $E(x)$. This vector is then quantized based on its distance to the prototype vectors in the codebook such that each vector $E(x)$ is replaced by the index of the nearest prototype vector in the codebook, and is transmitted to the decoder. The encoder network first transforms and downsamples the image, while a subsequent stack of residual blocks then further scales down the representations generating the latent map after quantization. The decoder is a feed-forward network that takes as input all levels of the quantized latent hierarchy and consists of a few residual blocks followed by a number of strided transposed convolutions to upsample the representations back to the original image size.

3.1 Custom solution and applied methodology

In the proposed solution we use a hierarchy of vector quantized codes to model large images, such that we are able to model local information, such as texture, separately from global information such as shape and geometry of objects. Therefore, our vanilla VQ-VAE based model consists in a multi-scale hierarchical encoder with a top latent code which models global information, and a bottom latent code, conditioned on the top latent, responsible for representing local details, and a prior model over each level which can is tailored to capture the specific correlations that exist in each level. The method consists in a two-stage approach: first, we train a hierarchical VQ-VAE to encode images (image compression) onto a discrete latent space, and then we fit a prior by means of an autoregressive neural network such as PixelCNN over the discrete latent space induced by all the data. For more detailed information about the proposed approach, please refer to the original paper[3].

stage 1 - VQ-VAE training We use a two level latent hierarchy where the encoder network first transforms the input image to a new latent representation which is then quantized into the top encoding, and the same operation is performed for the bottom encoding. Then we feed the decoder with the two previously found encodings to reconstruct the input image, we calculate the VQ-VAE loss and perform back-propagation.

stage 2 - Prior training The auxiliary prior is modeled with a autoregressive neural network such as PixelCNN in a second stage. The prior over the top latent map is responsible for structural global information, thus it is equipped with multi-headed self-attention layers. The prior is necessary in order to further compress the image, and to be able to sample from the model learned during the first stage. PixelCNN consists in residual gated convolution layers interspersed with causal multi-headed attention every five layers, regularized by incorporating dropout after each residual block as well as on the logits of each attention matrix, and with the addition of deep residual networks consisting of 1x1 point-wise convolutions on top of the model.

3.2 Experimental Results

To assess the performance of our methodology, we train a three level hierarchical model over the FFHQ dataset [9], and we use several metrics, such as negative log-likelihood and reconstruction error, classification accuracy score, and finally, FID and inception score.

Negative log likelihood metric Negative log likelihood (NLL) on the test and training allow us to analyze generalization and monitor over-fitting. The experimental result suggests us that neither the prior network nor the VQ-VAE overfit the training data.

Classification Accuracy Score According to Classification Accuracy Score (CAS) [10], by evaluating the classifier on VQ-VAE reconstructions of the test images we get a significant improvement compared to vanilla VQ-VAE, from 54.83 to 58.74 in top-1 accuracy.

4 Wasserstein GAN

In this final section we define a form of GAN (Generative Adversarial Network) called Wasserstein-GAN [4], and we describe the benefits of this new architecture with respect to the vanilla GANs, providing empirical evidence of its improvements in training performances and image generation. In order to explain Wasserstein-GAN, it is necessary to introduce the Wasserstein-1 distance, also called Earth-Mover (EM) distance, indicating how much "mass" must be transported from a former probability distribution to anther in order to transform the former in the latter one.

Earth-Mover (EM) distance (Wasserstein-1 distance) Earth Mover (EM) distance is defined as $W(P_r, P_g) = \inf_{\gamma \in i(P_r, P_g)} E_{(x,y)\sim\gamma}\big[||x-y||\big]$, where X is a compact metric set, $P_r, P_g \in Prob(X)$ are two probability distributions, $\gamma(P_r, P_g)$ denotes the set of all joint distributions $\gamma(x, y)$ whose marginals are respectively P_r and P_g. Intuitively, $\gamma(x, y)$ indicates how much "mass" must be transported from x to y in order to transform the distributions P_r into the distribution P_g. The EM distance then is the "cost" of the optimal transport plan.

Wasserstein-GAN We can divide the Wasserstein-GAN architecture in two different components: the critic (discriminator neural network) denoted with $f_w(.)$ with parameters w, and the generator (generator neural network) denoted with $g_\theta(.)$ with parameters θ. The generator gradient, to update the generator parameters θ is calculated as $g_\theta \leftarrow -\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$, where m is the batch size, $z^{(i)}$ is sampled generated data and g_w is the computed gradient; finally, we use RMSProp to calculate the parameters update. For each generator update we perform n_{critic} updating steps of the discriminator. However, the infimum of the Earth-Mover distance we introduced before is highly intractable, thus we calculate the gradient to update the critic's parameters with the modified objective $g_w \leftarrow \nabla_w \left[\frac{1}{m} \sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m} \sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$, where m is the batch size, $x^{(i)}$ is sampled real data, $z^{(i)}$ is sampled generated data, w are the critic's parameters, and g_w is the computed gradient; we use RMSProp optimizer to calculate the parameters update, and finally we perform clipping on the critic's parameters.

5 Experimental results

Wasserstein-GAN is proved to introduce two main important improvements with respect to standard GANs, consisting in improved stability of the optimization process, and improved generator convergence and sample quality. Our experimental results show that Wasserstein-GANs are much more robust than GANs when one varies the architectural choices for the generator, and it allows us to train the critic till optimality. When the critic is trained to completion, it simply provides a loss to the generator that we can train as any other neural network. This tells us that we no longer need to balance generator and discriminator's capacity properly. The better the critic, the higher quality the gradients we use to train the generator.

6 References

[1] Alexander, J.A. & Mozer, M.C. (1995) Template-based algorithms for connectionist rule extraction. In G. Tesauro, D.S. Touretzky and T.K. Leen (eds.), *Advances in Neural Information Processing Systems 7*, pp. 609–616. Cambridge, MA: MIT Press.

[2] Rifai, Salah, et al. "Contractive auto-encoders: Explicit invariance during feature extraction." Proceedings of the 28th International Conference on Machine Learning (2011).

[3] Razavi, Ali, et al. "Generating diverse high-fidelity images with VQ-VAE-2." Advances in Neural Information Processing Systems (2019).

[4] Arjovsky, Martin, et al. "Wasserstein GAN".

[5] Baldi, P. and Hornik, K. (1989). Neural networks and principal component analysis: Learning from examples without local minima. Neural Networks, 2, 53– 58.

[6] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. CoRR, abs/1312.6114, 2013.

[7] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. arXiv preprint arXiv:1410.8516, 2014.

[8] Hugo Larochelle and Iain Murray. The neural autoregressive distribution estimator. In Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, pages 29–37, 2011.

[9] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. arXiv preprint arXiv:1812.04948, 2018.

[10] Suman Ravuri and Oriol Vinyals. Classification accuracy score for conditional generative models. arXiv preprint arXiv:1905.10887, 2019.