Knowledge Distillation and its Variations

Sandipan Mandal & Teekam Chand Mandan

Introduction

In this project, we have implemented Knowledge Distillation(KD) from [1] and tried some variations of it. We evaluated the performance of KD on MNIST, notMNIST and SVHN datasets. Finally we have also implemented the paper Fitnets : Hints for Thin Deep Nets[2] and evaluated it on MNIST dataset.

Knowledge Distillation

In KD, we train a large teacher network $\mathbf{T}$. We train our smaller student network $\mathbf{S}$ using a composite loss function consisting of two terms, one optimizing to improve label prediction accuracy, other to match pre-softmax logits with teacher $\mathbf{T}$. Loss function is given by :

$$L(W_S) = H(y_{true}, P_S) + \lambda H(P_T^\tau, P_S^\tau)$$

where H is the cross-entropy loss function, y_{true} is the true label, P_S is the softmax output of $\mathbf{S}$, P_T^τ, P_S^τ are respectively softmax output of $\mathbf{T}$ and $\mathbf{S}$ after dividing the corresponding logits by τ.

Basic Adversarial Knowledge Distillation

Here, we train a large teacher network $\mathbf{T}$ and smaller student network $\mathbf{S}$. Using $\mathbf{T}$ and $\mathbf{S}$, we train a discriinator network $\mathbf{D}$, which can distinguish if a logit is constructed from $\mathbf{T}$ or $\mathbf{S}$. We train our final student network $\mathbf{S}$ using a composite loss function consisting of two terms, one optimizing to improve label prediction accuracy, other to match pre-softmax logits with teacher $\mathbf{T}$. The only difference here is that the logit matching is done adversarially using $\mathbf{D}$. The loss function is given by :

$$L(W_S) = \alpha H(y_{true}, P_S) - \beta H(network_{true}, P_D)$$

where P_D is the softmax output of the disriminator network which is fed the student logit as input, $network_{true}$ gives whether the given logit comes from $\mathbf{T}$ or $\mathbf{S}$. During training, the logit always comes from $\mathbf{S}$. The other symbols are the same as earlier.

Knowledge Distillation using GANs

Here we generalize the approach we have taken earlier. It is based on GANs. Though it uses GANs, the loss function we used is different from the one used by[3]. In this approach instead of pre-training the discriminator, we train discriminator, $\mathbf{D}$ and student network (generator), $\mathbf{G}$ on the go much like GANs. We assume the logits generated from $\mathbf{T}$ as **real** sample and the one generated from $\mathbf{S}$ as **fake** sample. The loss function for $\mathbf{D}$ is same as vanilla GAN, but for $\mathbf{G}$ which also happens to be student network has an additional term in it.

$$L(W_D) = -\gamma[log(P_{real}(x)) + log(1 - P_{fake}(x))]$$
$$L(W_G) = \beta H(y_{true}, P_S) - \alpha log(P_{fake}(x))$$

where $P_{real}(x) = \sigma(D(T(x)))$ and $P_{fake}(x) = \sigma(D(G(x)))$, $\sigma()$ being sigmoid function.

Knowledge Distillation using Wasserstein GANs

This aproach is exactly the same as the one above, except that here we use Wasserstein GANs (WGANs) instead of vanilla GANs. As a result the loss function is slightly different. The loss functions are:

$$L(W_D) = -\gamma[f_{real}(x) - f_{fake}(x)]$$
$$L(W_G) = \beta H(y_{true}, P_S) - \alpha f_{fake}(x)$$

where $f_{real}(x) = D(T(x))$ and $f_{fake}(x) = D(G(x))$

Columns

By default, blocks are arranged in a single column. If you want multiple columns for your poster, you may use the `columns` environment. For example,

```
\begin{columns}
\column{.6}
\block{...}{...}
\column{.4}
\block{...}{...}
\block{...}{...}
\end{columns}
```

will create two columns of 60% and 40% the available width; spacing between successive columns is handled automatically. The block command(s) following `\column` are the blocks to go in that column. The number of columns is free to be chosen, but the relative widths must all be chosen. If the widths sum to less than 1, empty space will be seen on the right. If they sum to more than 1, the latter columns will be cut off.

Subcolumns

If you want to have an additional subdivision of columns inside a column, you may use the `\subcolumns` environment inside of a column environment. The functionality is similar to that of columns, but now the widths are relative to the width of the current column.

An example use of subcolumns is.

```
\begin{subcolumns}
\subcolumn{.6}
\block{...}{...}
\subcolumn{.4}
\block{...}{...}
\block{...}{...}
\end{subcolumns}
```

References

[1] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. "Distilling the knowledge in a neural network". In: *arXiv preprint arXiv:1503.02531* (2015).

[2] Adriana Romero et al. "Fitnets: Hints for thin deep nets". In: *arXiv preprint arXiv:1412.6550* (2014).

[3] Zheng Xu, Yen-Chang Hsu, and Jiawei Huang. "Learning Loss for Knowledge Distillation with Conditional Adversarial Networks". In: *arXiv preprint arXiv:1709.00513* (2017).