

Residual Learning, Attention Mechanism and Multi-tasks Learning Networks.

Zheng Shi

ISE, Lehigh Univ.

April 22, 2020

Outline

- 1 Residual Learning [1, 2]
- 2 Attention Mechanism [3]
- 3 Multi-tasks Learning [4, 5]

Outline

- 1 Residual Learning [1, 2]
- 2 Attention Mechanism [3]
- 3 Multi-tasks Learning [4, 5]

Motivation

Increasing network depth does not work by simply stacking layers together.

- *degradation*: with the network depth increasing, accuracy gets saturated and then degrades rapidly.

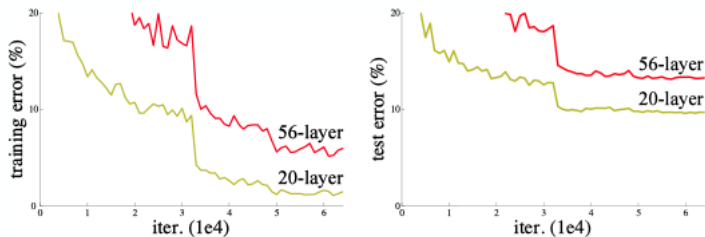


Figure 1: Source: [1]

Motivation

Unstable gradients in deep neural networks.

- *Vanishing gradient problem:*

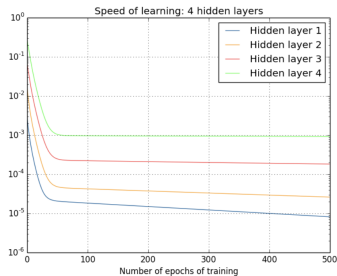


Figure 2: Source: <http://neuralnetworksanddeeplearning.com/chap5.html>

Method

Residual learning block

- *Skip connection:*

$$\mathbf{y} = h(\mathbf{x}) + \mathcal{F}(\mathbf{x}, \mathbf{W}), \quad (1.1)$$

$$= h(\mathbf{x}) + \mathbf{W}_2 \sigma(\mathbf{W}_1 \mathbf{x}). \quad (1.2)$$

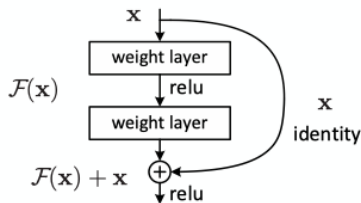


Figure 3: Source: [1]

Method

Assume h is an identity map, denote l the layer(s) of a residual learning block, \mathbf{x}_l is its input and \mathbf{x}_{l+1} is the output. Then, by Eq.1.1,

- *Forward pass*

$$\mathbf{x}_{l+1} = \mathbf{x}_l + \mathcal{F}(\mathbf{x}_l, \mathbf{W}_l) \quad (1.3)$$

Recursively, let L be any deeper/later layer with l be any shallower/early layer, we have

$$\mathbf{x}_L = \mathbf{x}_l + \sum_{i=l}^{L-1} \mathcal{F}(\mathbf{x}_i, \mathbf{W}_i) \quad (1.4)$$

Method

Carrying over the notations and denote the loss function as E , by chain rule, we have

- *Backward pass*

$$\frac{\partial E}{\partial \mathbf{W}_l} = \frac{\partial E}{\partial \mathbf{x}_L} \frac{\partial \mathbf{x}_L}{\partial \mathbf{x}_l} = \frac{\partial E}{\partial \mathbf{x}_L} \frac{\partial}{\partial \mathbf{W}_l} \sum_{i=l}^{L-1} \mathcal{F}(\mathbf{x}_i, \mathbf{W}_i). \quad (1.5)$$

- *What the author had in [2] is*

$$\frac{\partial E}{\partial \mathbf{x}_l} = \frac{\partial E}{\partial \mathbf{x}_L} \frac{\partial \mathbf{x}_L}{\partial \mathbf{x}_l} = \frac{\partial E}{\partial \mathbf{x}_L} + \frac{\partial E}{\partial \mathbf{x}_L} \frac{\partial}{\partial \mathbf{x}_l} \sum_{i=l}^{L-1} \mathcal{F}(\mathbf{x}_i, \mathbf{W}_i). \quad (1.6)$$

In Eq.1.6, $\frac{\partial E}{\partial \mathbf{x}_l}$ is decomposed into two additive terms: a term of $\frac{\partial E}{\partial \mathbf{x}_L}$ that propagates information directly without concerning any weight layers, and another term that propagates through the weight layers. It ensures that information is directly propagated back to any l and suggests that it is unlikely for the gradient to be canceled out for a mini-batch [2].

Empirical Results

Plain v.s. residual networks

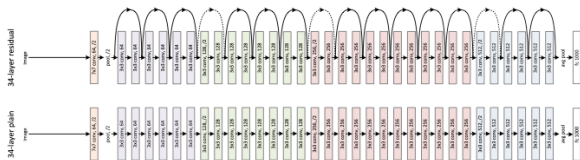


Figure 4: Source: [1]

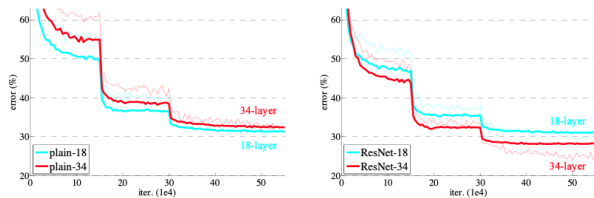


Figure 5: Source: [1]

Outline

- 1 Residual Learning [1, 2]
- 2 Attention Mechanism [3]
- 3 Multi-tasks Learning [4, 5]

Background and Motivation

- Attention mechanism is well known as a technique for machine translation, natural language processing and so on in a RNN setting.
- It can be applied to any kinds of neural networks.
- We will go over the basic idea based on Graph Attention Network (GAT) [3], which operates on graph-structured data in a CNN setting. But, it can be applied to any types of learning tasks.
- One of the major claimed benefits is to focus learning on significant representations/features while maintain an appropriate level of complexity.
- Some other benefits that authors argued are efficiency, scalability, being capable of generalizing to unseen graphs.

GAT Architecture

- *Input feature representation*

The input to the layer is a set of node features, denoted $\mathbf{h} = \{h_1, h_2, \dots, h_N\}$ with $h_i \in \mathbb{R}^F$, where N is the number of nodes and F is the number of features in each node.

- *Graph Attentional Layer*

A linear transformation, parameterized by $\mathbf{W} \in \mathbb{R}^{F' \times F}$ applied on h_i , to have a higher dimensional abstract representation, denoted h'_i ,

$$h'_i = \mathbf{W}h_i \quad \forall i \in \{1, 2, \dots, N\}. \quad (2.1)$$

Apply attention map, $a : \mathbb{R}^{F'} \times \mathbb{R}^{F'} \mapsto \mathbb{R}$, to obtain the attention coefficients with respect to nodes i, j ,

$$\alpha_{ij} = a(h'_i, h'_j) \quad (2.2)$$

GAT Architecture

- *Graph Attentional Layer* (continued)

In some sense, Eq.2.2 suggests the relative importance of node j 's features to node i . It might be desirable to restrict the index set of j , such that α_{ij} is only computed for j within a certain pre-defined neighborhood, denoted \mathcal{N}_i . Then, to normalize them across all $j \in \mathcal{N}_i$ using the softmax function,

$$\beta_{ij} = \frac{\exp(\alpha_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(\alpha_{ik})}. \quad (2.3)$$

- *Attention function*

In Eq.2.2, let a be a fully-connected layer parameterized by a weight vector $\mathbf{a} \in \mathbb{R}^{2F'}$ with some activation function, denoted σ , and then Eq.2.2 can be rewritten in the form

$$\alpha_{ij} = \sigma(\mathbf{a}^T [h'_i \parallel h'_j]), \quad (2.4)$$

where \parallel is a concatenation operation.

GAT Architecture

- *Output feature representation*

Provided β_{ij} for $j \in \mathcal{N}_i$, with a layer specific activation function, g , and transformed feature representation, h'_i , the output of the layer, denoted h''_i , is

$$h''_i = g\left(\sum_{j \in \mathcal{N}_i} \beta_{ij} h'_j\right). \quad (2.5)$$

- *Multi-head attention*

Given M independent attention functions, each of which is denoted by a^m for $m \in \{1, 2, \dots, M\}$, a multi-head attention generates the output as

$$h''_i = \parallel_{m=1}^M g\left(\sum_{j \in \mathcal{N}_i} \beta_{ij}^m h_j^{m'}\right). \quad (2.6)$$

- *Multi-head attention* (continued)

The output of the last hidden layer of the network, is an average of M attentions.

$$h_i'' = g\left(\frac{1}{M} \sum_{m=1}^M \sum_{j \in \mathcal{N}_i} \beta_{ij}^m h_j^{m'}\right), \quad (2.7)$$

where

$$\begin{aligned} h_i^{m'} &= \mathbf{W}^m h_i \\ \forall i &\in \{1, 2, \dots, N\}, \\ \forall m &\in \{1, 2, \dots, M\}. \end{aligned} \quad (2.7)$$

GAT Architecture

- Illustration on single and multi-head attention mechanism

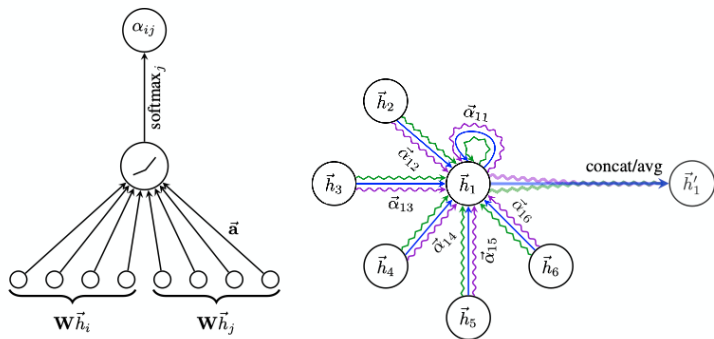


Figure 6: Source: [3]

Empirical Results

- Sample output of GAT

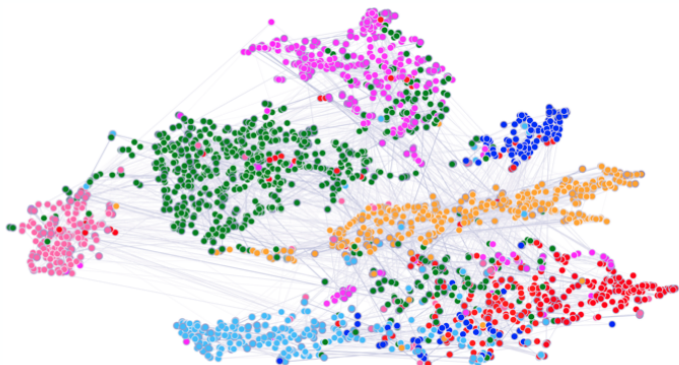


Figure 7: Source: [3]

Empirical Results

- Inductive Learning

<i>Inductive</i>	
Method	PPI
Random	0.396
MLP	0.422
GraphSAGE-GCN (Hamilton et al., 2017)	0.500
GraphSAGE-mean (Hamilton et al., 2017)	0.598
GraphSAGE-LSTM (Hamilton et al., 2017)	0.612
GraphSAGE-pool (Hamilton et al., 2017)	0.600
GraphSAGE*	0.768
Const-GAT (ours)	0.934 ± 0.006
GAT (ours)	0.973 ± 0.002

Figure 8: Source: [3]

Empirical Results

- Transductive Learning

<i>Transductive</i>			
Method	Cora	Citeseer	Pubmed
MLP	55.1%	46.5%	71.4%
ManiReg (Belkin et al., 2006)	59.5%	60.1%	70.7%
SemiEmb (Weston et al., 2012)	59.0%	59.6%	71.7%
LP (Zhu et al., 2003)	68.0%	45.3%	63.0%
DeepWalk (Perozzi et al., 2014)	67.2%	43.2%	65.3%
ICA (Lu & Getoor, 2003)	75.1%	69.1%	73.9%
Planetoid (Yang et al., 2016)	75.7%	64.7%	77.2%
Chebyshev (Defferrard et al., 2016)	81.2%	69.8%	74.4%
GCN (Kipf & Welling, 2017)	81.5%	70.3%	79.0%
MoNet (Monti et al., 2016)	81.7 \pm 0.5%	—	78.8 \pm 0.3%
GCN-64*	81.4 \pm 0.5%	70.9 \pm 0.5%	79.0 \pm 0.3%
GAT (ours)	83.0 \pm 0.7%	72.5 \pm 0.7%	79.0 \pm 0.3%

Figure 9: Source: [3]

Outline

- 1 Residual Learning [1, 2]
- 2 Attention Mechanism [3]
- 3 Multi-tasks Learning [4, 5]

Background and Motivation

- Deep learning applications in some engineering fields often have objectives to perform a few related tasks with a single model or an ensemble of models and can sometimes share the same dataset.
- Creating auxiliary tasks can potentially improve the learning performance on the desired task.
- We can view multi-task learning (MTL) as a form of inductive transfer, which can help improve a model by introducing inductive bias and cause models to prefer some hypothesis over others.

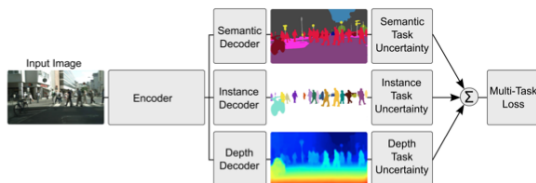


Figure 10: Source: [4]

Methods

- *Hard parameter sharing*

In general, it makes all tasks share the hidden layers and, sometimes, keep several task-specific output layers. Intuitively, with respect to the original task, multiple tasks restricts the function that the network model approximates and thus reduce the risk of overfitting.

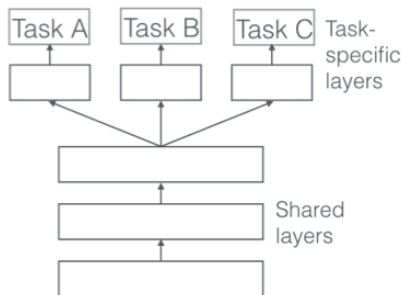


Figure 11: Source: [4]

Methods

- *Soft parameter sharing*

Each task has its own model with its own parameters. However, constraints are imposed to enforce the proximity of parameters between models.

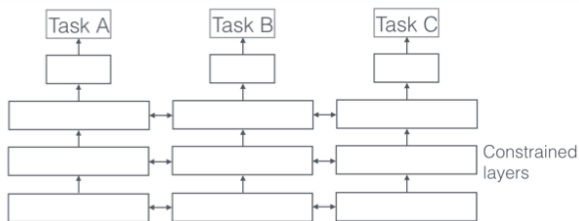


Figure 12: Source: [4]

Auxiliary Tasks

With suitable auxiliary tasks, MTL can be a good tool to boost the performance on the original task.

- *Related task*

For instance, tasks that predict different characteristics of the road as auxiliary tasks for predicting the steering direction in a self-driving car; use head pose estimation and facial attribute inference for facial landmark detection; jointly learn query classification and web search.

- *Hints*

Having tasks serving as references or indicators for intermediate output or final output of the model. For instance, predicting whether an input sentence contains a positive or negative sentiment word as auxiliary task for sentiment analysis; predicting whether a name is present in a sentence for name error detection.

Auxiliary Tasks





- *Focusing attention*

Having tasks to reinforce a learning process on certain characteristics. For instance, for learning to steer a self-driving car, a single-task model might ignore lane markings as they only make up a small part of the image or are not always present. Predicting lane markings as auxiliary task could force the model to learn the represent them, which could be useful for the main task.


- *Representation learning*

The goal of an auxiliary task in MTL is to enable the model to learn representations that are shared or helpful for the main task. All auxiliary tasks discussed so far do this implicitly: They are closely related to the main task, so that learning them likely allows the model to learn beneficial representations. A more explicit modelling is possible, for instance by employing a task that is known to enable a model to learn transferable representations.

Bibliography I

-  Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun.
Deep residual learning for image recognition.
arXiv preprint arXiv:1512.03385, 2015.
-  Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun.
Identity mappings in deep residual networks.
arXiv preprint arXiv:1603.05027, 2016.
-  Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio.
Graph attention networks.
arXiv preprint arXiv:1710.10903, 2017.
-  Sebastian Ruder.
An overview of multi-task learning in deep neural networks.
arXiv preprint arXiv:1706.05098, 2017.

Bibliography II

-  Yu Zhang and Qiang Yang.
A survey on multi-task learning.
arXiv preprint arXiv:1707.08114, 2017.