Neural ODEs for Machine Translation

Enrique De Alba
edealba@stanford.edu

Glib Stronov
gstronov@stanford.edu

OVERVIEW

Introduction: Despite recent successes in the field of machine translation, there are still several problems that exists pertaining to the quality of the translations produced. We attempt to adapt the concept of Neural Ordinary Differential Equations to the task of machine translation. We show that using ODESolvers to solve differential equations as a new proxy for forward functions in an NMT context performs on par with traditional LSTM-based NMT systems.

Motivation: Because traditional neural network models tend to either cap off or perform slightly worse after a certain number of layers, it may be possible that using Neural ODE systems to integrate over an arbitrary amount of layers can outperform traditional models given an inordinate amount of training time.

APPROACH

Residual Networks and Recurrent Neural Network decoders build complex transformations on some data x by sequentially composing transformations to define the next hidden state h_t. For Residual Networks we have the following transformation: $h_t = x_t + F_t(x_t)$ where F_t is the t^{th} hidden layer. We can differentiate h as a function of t as the step size $\epsilon \to 0$ (which was previously 1). Here, t represents a continuous analog for the network's layers. We can then define the hidden layer $h(T)$ as the solution to the following integral:

$$h(T) = \int_0^T \frac{dh(t)}{dt}\, dt = \int_0^T \mathcal{F}\big(h(t)\big)\, dt$$

Just like with traditional neural networks, we would need to figure out a way to perform backpropagation to improve our model. In the continuous case, however, it is not as clear-cut, since we no longer have discrete transitions between layers. In order to address this issue, we present a way to perform a continuous method for backpropagation.

$$L\big(h(t_1)\big) = L\left(h(t_0) + \int_{t_0}^{t_1} \mathcal{F}\big(h(t)\big)\, dt\right) = L\Big(\text{ODESolver}\big(h(t_0), \mathcal{F}, t_0, t_1, \theta\big)\Big)$$

$$\frac{\partial L}{\partial \theta} = \int_{t_1}^{t_0} -a(t)^{\top} \frac{\partial \mathcal{F}\big(h(t)\big)}{\partial \theta}\, dt$$

$$\frac{\partial L}{\partial h(t_0)} = \frac{\partial L}{\partial h(t_1)} + \int_{t_1}^{t_0} -a(t)^{\top} \frac{\partial \mathcal{F}\big(h(t)\big)}{\partial h}\, dt = a(t_0)$$

References:
Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. *Neural Ordinary Differential Equations*. University of Toronto, Vector Institute, 2019
Marco Ciccone, Marco Gallieri, Jonathan Masci, Christian Osendorfer, and Faustino Gomez. NAIS-NET: *Stable Deep Networks from Non-Autonomous Differential Equations*. NNAISENSE SA, 2018
Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. *Character-Aware Neural Language Models*. Cornell University, 2015
Yiping Lu, Aoxiao Zhong, Quanzheng Li, and Bin Dong. Beyond finite layer neural networks: *Bridging deep architectures and numerical differential equations*. Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden
Lars Ruthotto and Eldad Haber. *Deep Neural Networks Motivated by Partial Differential Equations*. arXiv preprint arXiv:1804.04272, 2018

EXPERIMENTS

We utilize an ODEBlock and input the convolutional network's output $x_{convout}$ into it to output the tuple ($x_{convout}$, ODESolver(F, $x_{convout}$, t_{int})) where F is the ODEFunction of the ODEBlock and takes $x_{convout}$ as its input and t_{int} is the integration time used in the ODESolver. The ODEBlock outputs ODESolver(F, $x_{convout}$, t_{int}) which goes through a dropout layer and outputs the final word embeddings subsequently used in the Seq2Seq translation network.

NODE Vanilla ResBlock Forward Function:

$$a \leftarrow \text{ReLU}\left(W_i x + b_i\right) \text{ where } W_i \in \mathbb{R}^{e_{\text{word}} \times e_{\text{word}}}, b_i \in \mathbb{R}^{e_{\text{word}}}$$

$$\mathcal{F}_{\text{baseline}}(t, x) = x + \text{ReLU}\left(W_j a + b_j\right) + a$$

NODE-t Concatenated Forward Function:

$$X^* \leftarrow \text{ReLU}(x)$$

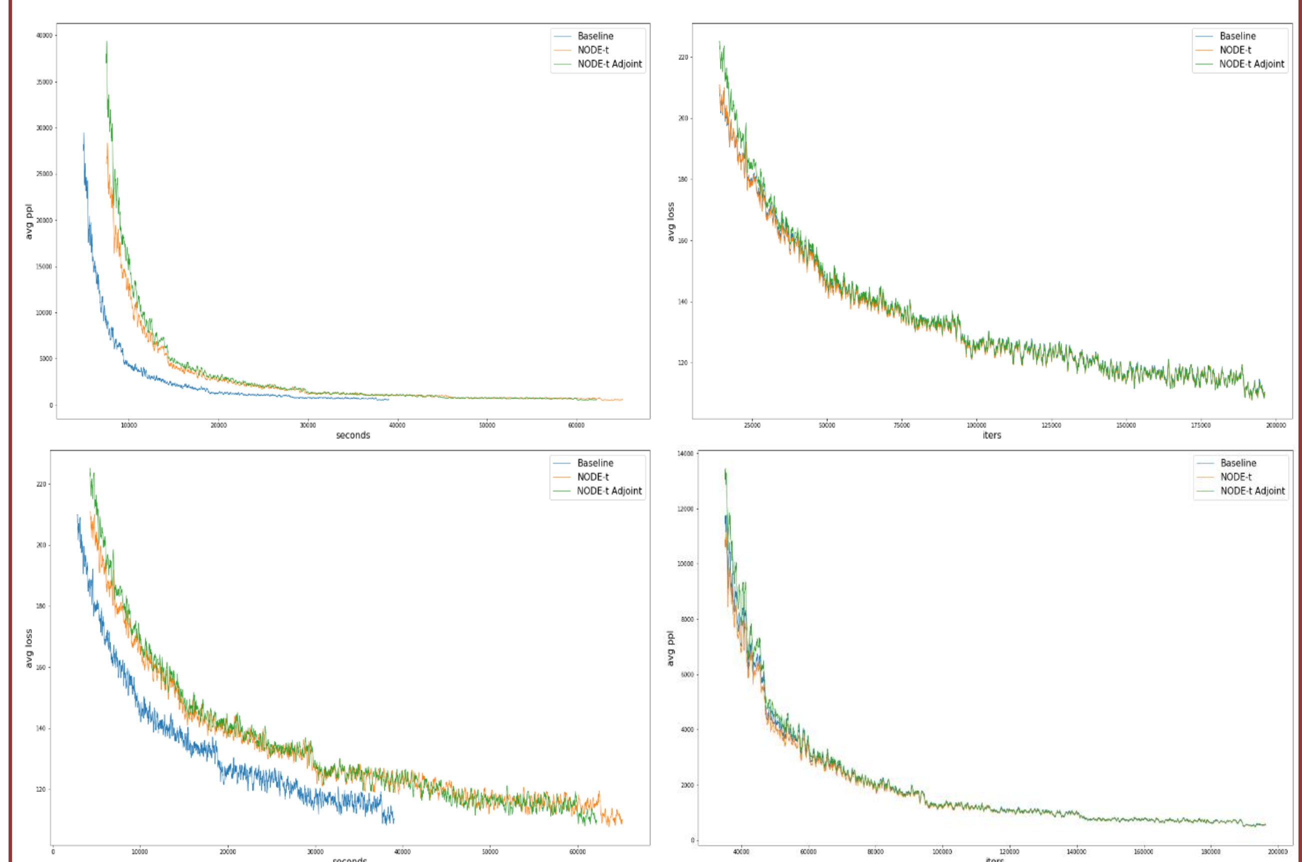
$$A^* \leftarrow \text{ReLU}\left(W_{t(1)} T + b_{t(1)}\right) \text{ where } W_{t(1)} \in \mathbb{R}^{e_{\text{word}}+1 \times e_{\text{word}}}, b_{t(1)} \in \mathbb{R}^{e_{\text{word}}}$$

$$\mathcal{F}_{\text{t-concat}}(t, x) = x + W_{t(2)} \widetilde{T} + b_{t(2)} \text{ where } W_{t(2)} \in \mathbb{R}^{e_{\text{word}}+1 \times e_{\text{word}}}, b_{t(2)} \in \mathbb{R}^{e_{\text{word}}}$$

$$\text{where } T = \left[\begin{array}{c|c} t & \\ \vdots & \mathbf{X}^* \\ t & \end{array}\right] \text{ and } \widetilde{T} = \left[\begin{array}{c|c} t & \\ \vdots & \mathbf{A}^* \\ t & \end{array}\right]$$

RESULTS

	BLEU	Train Time (hrs)	Decoding Time (mins)	Avg Words/Sec
Baseline	24.45	10.8	9.2	2854
NODE Vanilla adj	24.05	25.2	58.3	
NODE$-t$	24.25	18.1	96.6	1716
NODE$-t$ adj	24.45	17.3	71.4	1791



TRANSLATION CASES

Human Reference	In education, the one thing we know how to measure best is IQ.
Baseline	In education.
NODE Vanilla adj	In education, the only thing we know how to measure a better way.
NODE$-t$	In education, the only thing we know how to measure in better ways.
NODE$-t$ adj	In education, the only thing we know how to measure the best way.

Human Reference	First, botanical – which you can kind of get a sense of.
Baseline	First of all, bottles.
NODE Vanilla adj	First, biologists – that you can get a idea.
NODE$-t$	First, buttons that will be able to make an idea.
NODE$-t$ adj	First, bottoms, you can make an idea.

Human Reference	Your niece?
Baseline	Is it the one to your owner?
NODE Vanilla adj	Her supernova? Is her friend?
NODE$-t$	Is it your support? Is your friend of friend?
NODE$-t$ adj	Her supply to his friend?

In the first example, we see that all the NODE MT models produce longer sentences that are much closer to the human reference, and given the original Spanish sentence, as close as the translations could get to the human reference.
In the second example we see the baseline producing a very short and very incorrect translation - translating "botanical" as "bottles" and not mentioning anything else. The NODE models also have similar troubles translating "botanical" but do a better job in the second half of the sentence.
The third example shows a case where all models have incredible difficulty translating the source sentence.

CONCLUSION

We show that using ODESolvers to solve differential equations as a new proxy for forward functions in an NMT context performs on par with traditional LSTM-based NMT systems. These NODE machine translation models display significantly worse training/testing speeds compared to the traditional baseline NMT model because it is difficult to compete with respect to speed due to the rigorous optimization that has gone into LSTMs compared to ODESolvers. Integrating forward differential equations is an exciting new paradigm that may prove to be more effective when increasing the depth of ResBlock layers and integrating these layers as functions of t. NODE models should in theory be able to produce arbitrarily deep ResBlock layers, so despite of their speed inefficiencies, may still end up producing better models given enough time.

Future Work:

Experiment with increasing depth of ResBlock layers, including experimenting with setting the depth as a function of t.

Explore how changing the error tolerance of the ODESolver could speed up training, etc and how changing other hyperparameters may affect performance.