RESIDUAL NETWORKS CLASSIFY INPUTS BASED ON THEIR NEURAL TRANSIENT DYNAMICS

Anonymous authors
Paper under double-blind review

ABSTRACT

In this study, we analyze the input-output behavior of residual networks from a dynamical system point of view by disentangling the residual dynamics from the output activities before the classification stage. For a network with simple skip connections between every successive layer, and for logistic activation function, and shared weights between layers, we show analytically that there is a cooperation and competition dynamics between residuals corresponding to each input dimension. Interpreting these kind of networks as nonlinear filters, the steady state value of the residuals in the case of attractor networks are indicative of the common features between different input dimensions that the network has observed during training, and has encoded in those components. In cases where residuals do not converge to an attractor state, their internal dynamics are separable for each input class, and the network can reliably approximate the output. We bring analytical and empirical evidence that residual networks classify inputs based on the integration of the transient dynamics of the residuals. Different inputs are considered as different initial conditions that undergo different transitions through the network, and finally end up in different representations in the output layer. These transitions are critical in assigning the right class to the data. Based on these findings, we also develop a new method to adjust the depth for residual networks during training. As it turns out, after pruning the depth of a Resnet using this algorithm,the network is still capable of classifying inputs with a high accuracy.

1 INTRODUCTION

Residual networks (Resnets), first introduced in He et al. (2016), have been more successful in classification tasks in comparison with many other standard methods. This success is attributed to the skip connections between layers that facilitate the propagation of the gradient throughout the network, and in practice allow very deep networks to undergo a successful training. Apart from mitigating the gradient problem in deep networks, the skip connections introduce a dependency between variables in different layers that can be seen as a system state. This novelty provides an opportunity for interesting theoretical analysis of their functioning, and has been the underlying pillar for some interesting analysis of such networks from dynamical system point of view Ciccone et al. (2018); Chang et al. (2018); Haber & Ruthotto (2017); Lu et al. (2017); No & Liao (2016); Ruthotto et al. (2018); Chaudhari et al. (2017). In Haber & Ruthotto (2017), the authors have studied residual networks using difference equations, and analyzed the stability of the forward propagation of input data, and have linked the inverse problem to the well-posedness of the learning problem. To circumvent the vanishing or exploding gradient problem, it is suggested in Haber & Ruthotto (2017) to design the eigenvalues of the feedforward propagation close to the edge of stability, so that the inverse problem is not ill-posed. It is however not clear whether having the eigenvalues set close to the edge of stability is beneficial for the network performance, because it depends on the dynamics required by the actual task. Employing this idea, the authors in Chang et al. (2018) suggest a new reversible architecture for neural networks based on Hamiltonian systems. Also, in a recent study, Resnets have been employed as an unrolled non-autonomous time-invariant (with weight sharing) system of differential equations Ciccone et al. (2018), wherein, each Resnet block receives an external input, which depends on the previous block. This successive process of feeding the following block by the output of the previous block continues until the latent space converges. Our approach in this paper is similar to the aforementioned studies, however, to understand the classification mechanism

in Resnets, we focus on the role of the intrinsic transient dynamics of the residuals over different layers of the network.

Some studies on Resnets have focused on tracking the features layer by layer Greff et al. (2017); Chu et al., and have challenged the idea that deeper layers in neural networks build up abstract features that are different than those formed in lower layers. One supporting evidence for this challenge comes from lesion studies on Resnets Veit et al. (2016) and Highway networks Srivastava et al. (2015) which show that after the network is trained, perturbing the weights in the deep layers does not have a fundamental effect on the network performance, and therefore, does not bring the performance to chance level. However, changing the weights which are closer to initial layers, have more damaging effect. Empirical studies in Greff et al. (2017); Chu et al. suggest an alternative explanation for feature formation in deep layers; that is, successive layers estimate the same features which, along the depth of the network, are more refined, and yield an estimate with smaller standard deviation than earlier layers. Our study supports this idea by showing that features in different layers of a Resnet with shared weights are formed by the transient dynamics of residuals that may converge towards their steady state values if they are stable. In attractor networks, perturbing the initial layers changes those dynamics more drastically compared to perturbation of deeper layers, because the residuals in the deep layers are either very close to their stable fixed point, or have already converged. If there are no attractors for the residuals, sensitivity to initial conditions and the internal dynamics of the residuals play an important role in classification. In this case, perturbations of the network at initial layers can potentially change the dynamic evolution of the residuals completely, and this will have a more sever impact on the output. Classification based on unstable internal states is similar to Reservoir networks Maass et al. (2002), where it has been shown that the high dimensionality of the neurons at the readout layer can compensate for the lack of stability of the neural activities.

Moreover, one important topic in this domain is the depth of Resnets. On the one hand, the success of Resnets in classification has been attributed to their deep architecture He et al. (2016), on the other hand, there are studies that claim most of the training is accomplished in the initial layers, and having a very deep architecture is not necessary Zagoruyko & Komodakis (2016). Another challenge is to understand the generalization property of Resnets (which may be related to its depth), because their power is correlated with their ability in recognizing unseen data that also belong to the classes that these networks have been trained on. In our analysis of the transient and steady state dynamics of the residuals, we discuss these issues. In fact, an important finding of this paper is that residual networks classify inputs based on summing over all the residual's outputs throughout the network, meaning different transitions of residuals (convergence to their steady state, or their long wandering trajectories without convergence) can potentially change the classification result. Interestingly, in biological neuronal networks, it has been suggested that optimal stimulus separation in neurons that encode sensory information occurs during the transients rather than the fixed points of the neuronal activity trajectories Mazor & Laurent (2005); Rabinovich et al. (2008). Also, it has been discussed that spatio-temporal processing in cortical circuits are state dependent, and the role of transients are crucial Buonomano & Maass (2009). In our study we show that also in Resnets, these transients are the decisive factors for classification. Based on this finding, we develop a new method to control the depth of Resnets.

This study mainly emphasizes the importance of internal transient dynamics in Resnets on classification performance. Using a dynamical system approach, for a general Resnet, we derive dynamics of state evolutions of the residuals in different layers. Particularly, we show how these dynamical variables cooperate or compete with each other to build a common representation of the input classes in a network with shared weights between layers. It is well-known that very deep residual networks with weight sharing are equivalent to shallow recurrent neural networks, with similar performance to Resnets with variable weights between layers No & Liao (2016). Inspired by this work, we study Resnets with shared weights and sigmoid activation functions, which provide a more tractable mathematical analysis. Then, we show empirically that those dynamics are observed in Resnets with variable weights as well. We explain that Residual networks, in general, encode the information about the observed inputs in the dynamics of the residuals, such that stable steady-state values of the residuals, upon their existence, represent the common features among all input data. Moreover, we explain analytically and empirically that in very deep networks, residuals do not necessarily converge to zero. This study also gives examples of networks with either one or multiple stable or metastable (saddle) fixed points of the residuals, which shows that in general, the number of fixed points that play role in classification depends on the data and the network conditions.

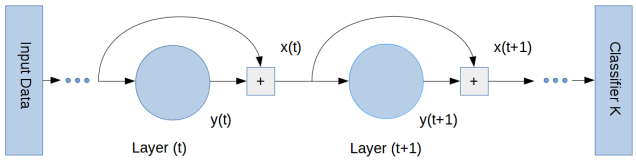

Figure 1: A simple schematic of skip connections between two successive layers. The dimensionality of the network does not change with depth. The variable $\mathbf{y}(t)$ represents the values of the residuals at layer t, and $\mathbf{x}(t)$ is the activation of neurons at layer t.

The main contributions of the paper are:

1. We show that Resnets classify input patterns based on the sum of the transient dynamics of the residuals in different layers.
2. For a network with shared weights between layers, we derive interaction dynamics between residuals and their state evolution.
3. Based on residual dynamics, we develop a new method to obtain an adaptive depth for Resnets, during training, for input classification.

2 Dynamics of Interactions Between Residuals

We consider a dense Resnet with N input dimensions, and arbitrary T layers with exactly N neurons at each layer. A unique property of a Resnet that distinguishes it from conventional feedforward networks is the skip connection between layers. In the Resnet we consider here, the activity of neuron i at layer t, before the output of the previous layer $(t-1)$ is added to it, is represented as $y_i(t)$, and the activity of all neurons in the same layer is represented by the vector $\mathbf{y}(t)$. After the integration of the output from layer $t-1$, the output of layer t is represented by $\mathbf{x}(t)$. The components of these residuals $\mathbf{y}(t)$ are calculated based on a linear function of $\mathbf{x}(t)$, i.e. $z_i(t) = \sum_{i=1}^{N} w_{ij}(t)x_j(t)$ followed by a nonlinear function $f(z_i)$. Figure 1 illustrates the relation between $\mathbf{x}$ and $\mathbf{y}$. Any hidden layer t represents a sample of the dynamical states $\mathbf{x}$ after t steps. This implies that the network at different layers calculates samples of $\mathbf{x}(t)$. Input data is considered as the initial condition of the system, and is depicted by $\mathbf{x}(0)$.

Interpreting the network as a dynamical system which evolves throughout the layers, the dynamics of neural activations are $\mathbf{x}(t+1) = \mathbf{x}(t) + \mathbf{y}(t+1)$, where $\mathbf{y}(t)$ is the output of the neurons, and in the rest of the paper, they are called "residuals". This equation implies a difference equation for the variable $\mathbf{x}(t)$, that is $\mathbf{x}(t+1) - \mathbf{x}(t) = \mathbf{y}(t+1)$. The left side of this equation resembles the forward Euler method of derivative of a continuous system, when the discretization step is equal to 1. This approximates a continuous system with dynamics that follow $\dot{x}_i(t) = y_i(t)$.

We are interested in understanding how the neural activities evolve over layers in a feedforward fully connected network. It is easier to study the underlying dynamics in the continuous-time version of the system, assuming that the neuronal activities were samples of the original system, resulting from the forward difference Euler method. With this assumption, the dynamics of $\mathbf{x}(t)$ and inputs to the residuals $\mathbf{z}(t)$ follow

$$\begin{aligned} \dot{x}_i(t) = y_i(t) = f(z_i(t)) &\Longrightarrow x_i(t) = \int_0^t y_i(\tau)d\tau + x_i(0) \\ z_i(t) = \sum_{j=1}^{N} w_{ij}(t)x_j(t) + b_i &\Longrightarrow \dot{z}_i(t) = \sum_{j=1}^{N} w_{ij}(t)\dot{x}_j(t) + \dot{w}_{ij}(t)x_j(t) \end{aligned} \tag{1}$$

The first line of equation 1 indicates that $\mathbf{x}(t)$ stores the sum of the residuals and the input ($\mathbf{x}(0)$) from the input layer up to layer t, meaning that $\mathbf{x}(t)$ is a cumulative signal for $\mathbf{y}(t)$, as well as the input data. For a Resnet with shared weights between layers, $\dot{w}_{ij}(t) = 0$ because the weights do not change between layers (Note that the variable t corresponds to the layer indexed by t, and because

there are no changes between weights in different layers, the derivative of the weight dynamics across layers is equal to zero, by design). This constraint makes the analysis simpler, and results in $\dot{z}_i(t) = \sum_{j=1}^{N} w_{ij}(t)\dot{x}_j(t)$. In this case, after replacing $\dot{x}_i(t)$ by $y_i(t)$ in equation 1, the dynamics of $\mathbf{z}(t)$ and the residuals $\mathbf{y}(t)$ will be

$$\begin{aligned} \dot{z}_i(t) &= \sum_{j=1}^{N} w_{ij}(t) y_j(t) \\ \dot{y}_i(t) &= \frac{\partial y_i(t)}{\partial z_i(t)} \frac{\partial z_i(t)}{\partial t} = \frac{\partial f(z)}{\partial z} (\sum_{j=1}^{N} w_{ij}(t) y_j(t)). \end{aligned} \tag{2}$$

The last equation indicates that the dynamics of the residuals is a product of two terms: the derivative of the activation function with respect to its own input, and a linear combination of all other residuals at the same layer. A steady state solution for y_i, in a network with a time invariant weight w_{ij} (shared weights across layers), is obtained from setting either of these two terms to zero. For a general nonlinear $f(.)$, according to equation 2, the interactions between the residuals are either competitive or cooperative, depending on the positive or negative influence that they have on each other. For activation functions which their derivatives are in the form of $\dot{\mathbf{y}} = \mathbf{y}(\mathbf{G}(\mathbf{y}))$, where $\mathbf{G}(.)$ is a linear or nonlinear function, the dynamics of the residuals follow a predator-prey type of equations.

The dynamical system interpretation of Resnets clarifies that perturbations at initial layers can change the dynamic trajectory of the residuals more drastically. According to equation 1 and figure 1, the cumulative of the residuals over the entire network feeds the classifier. Therefore, those perturbations disrupt the output more severely. A discrete-time analysis of the effect of perturbations at different layers on the output value is given in the Appendix.

2.1 Residuals for a sigmoid activation function

For a *logistic* function $f(.)$, the derivative is $y_i(t)(1 - y_i(t))$, which results in a particular form of predator-prey equation, well-known in studying ecosystems. In this case, equation 2 yields

$$\dot{y}_i(t) = y_i(t)(1 - y_i(t))(\sum_{j=1}^{N} w_{ij}(t) y_j(t)) \tag{3}$$

For the sake of simplicity, in our analysis, we studied a network with a time invariant W (shared weights between layers). Depending on the sign of w_{ij}, each y_j can have a positive or negative influence on the growth rate of y_i, hence plays the role of prey or predator. For N residuals, the number of possible fixed points (solutions of equation 3) is 3^N. However, from a dynamical system point of view, not all of them can be stable for a given weight W. The initial condition for the residuals is determined by the output of the first layer. After some transients over the next layers, each residual converges to its stable solution, if there is one. Note that the derivatives at $y_i = 0$ or 1 are equal to zero, and the system's trajectories are confined to this space. We hypothesize that for an attractor network, for a classification with a more generalization capacity, it is better to have the the steady state values of the residuals at the final layer of the network. The depth of the network is an important determining factor for each residual to end up in its equilibrium, or still stay in its transient regime. Other determining factors are the eigenvalues, and the initial conditions. For non-attractor networks, the residuals are always in the transient regime, however, if the transitions are long enough to yield separable states for each class, the network will be able to classify the input patterns (similar to classification using reservoir computing Maass et al. (2002)).

3 Experiments on low-dimensional data

We considered an illustrative example of two concentric circles with radius 0.5 and 1, each corresponding to a different class. Using a network with 3 neurons in each layer, and with 15 hidden layers of sigmoid activation functions with shared weights, 1000 training samples, and 1000 test samples, the classification accuracy on the test set was 100%.

As illustrated in figure 2 after training, the residuals converge to the stable fixed point of the system equation 3 which is $(y_1, y_2, y_3) = (0., 0., 1.0)$ (with eigenvalues $w_{13} = -1.95$, $w_{23} = -0.87$,

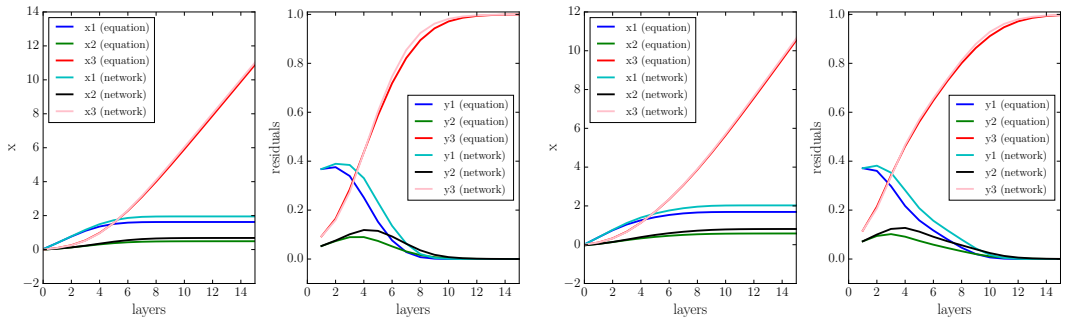

Figure 2: The mean of the residuals and the activities of the neurons at the last hidden layer (cumulative signal for the residuals) of a 15 layer Resnet are compared to the integration results of equation 3. Left: Input data are 1000 samples of a circle with radius 0.5. Right: Results for 1000 samples of a circle with radius 1.0

and $-w_{33} = -0.8$). This fixed point is identical for both classes. The neural activations ($\mathbf{x}(t)$) evolving across layers are monotonic, and in this case, the final value of $\mathbf{x}(T = 15)$ is different for the two classes (for radius = 0.5, $\mathbf{x}(15) = [1.95, 0.68, 12.02]$ while for radius = 1.0, $\mathbf{x}(15) = [2.03, 0.81, 11.63]$). This difference is due to different shapes of the residual curves in each case. Since the inner product of each of these two different vectors with the classifier ($K = [1.29, 5.11, -0.66]$) result in different numbers, the vector $\mathbf{x}$ which contains the cumulative sum of the residuals for each class, determines the classification outcome. This example implies that even if the residuals that result from two different classes of datasets converge to a single fixed point, because they have different transitions towards the fixed point, the cumulative of the residuals will have a different representation at the final hidden layer before the classifier. This signal together with the classifier determine the value of the output neuron.

4 Experiments on MNIST

To study the behavior of the network on large datasets, we considered a network with 1064 sigmoid neurons in each layer, and 15 layer deep. First, we analyzed a case where the weight matrix was shared between all the layers. The input data was chosen from MNIST, and the classification was performed by using the softmax algorithm. In this case, the classification error on the test set was 1.4%. Note that the network considered here is the simplest possible network architecture (so as to allow us to understand the classification mechanism in Resnets) with shared weights; therefore, the results are not comparable to the state of the art performance on MNIST. As illustrated in figure 3 top panel, the residuals in the first few layers are still in the transition period (non-zero standard deviation for 200 random samples from 10 classes). We used the same weights in a network with 1000 hidden layers (without retraining) to study the behavior of the residuals in a deep version of the same Resnet. There were a few non-zero fixed points with some negligible standard deviation among 200 samples. To check if the fixed points were different and distinguished for each class, we plotted the average and the standard deviation of the residuals for the test set, separately for each class, on the final hidden layer in figure 6 in the appendix. The average for each class is different from any other class, however, a large number of dimensions are identical. The standard deviation between the residuals in a single class are small, and negligible, and indicate that the classification accuracy is not 100%. For this prolonged simulation, we obtained the eigenvalue distribution for the average residuals for each class at layer 1000. Residuals corresponding to classes $0, 2, 3$ had a single small positive eigenvalue (around 0.02) among all other negative eigenvalues (saddle point). This means that those classes are still in their transition period at layer 1000, and due to the small value of the positive eigenvalue, the transition is slow.

Due to the high dimensionality of the network in this case, it is not viable to illustrate the transition dynamics of individual neurons separately for each class. However, to show different transition patterns of the residuals in each class, we chose one neuron for each class such that the classifier

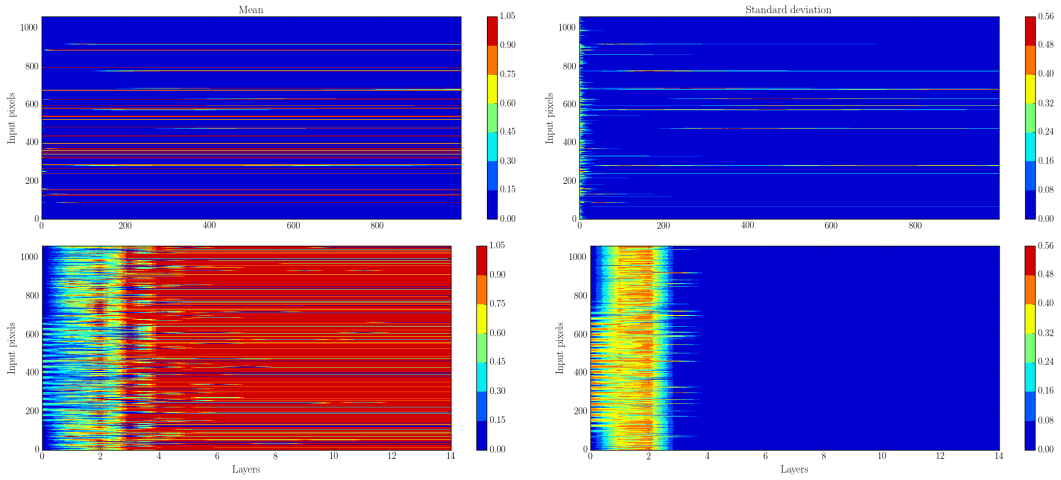

Figure 3: Mean and standard deviation of the residuals for all 10 classes for a 1000 layer deep Resnet with shared weights (top), and a 15 layer Resnet with variable weights (bottom). Top: The weights are from a 15-layer deep network. The stable residuals are sparse in activities. Bottom: The network with 15 different weight matrices has more residuals at their maximum values which correspond to many saturated neurons in the final layer. The standard deviation of the residuals are zero after the 4th layer, showing that the residuals have no transient dynamics in the following layers.

had the highest sensitivity to the value of the cumulative transitions of that neuron. The index of this neuron was derived from the sensitivity of each class C with respect to $\mathbf{x}(T)$, which is the classifier K. For each output in the softmax layer, there exists a maximally sensitive weight for its corresponding classifier K. This method renders 10 different indices. In figure 4, we plotted the average (over 1000 samples for each class) of the residuals for neurons that corresponded to those indices. In almost all cases (apart from class 8), the maximum value of $x(15)$ belonged to the neuron that had the largest coefficient in the classifier vector for that particular class. This implies that separation between classes are encoded in the transient dynamics of those neurons and other neurons that their cumulative trajectories are multiplied by big coefficients in the classifier. The transient dynamics of those neurons play an important role in the classification result.

To check the classification behavior of the network, we visualized the activities of all neurons at the final hidden layer $x(15)$ for all 10 different classes in MNIST, in terms of the mean and standard deviations across samples which belonged to each class. Figure 5 illustrates that the mean of the outputs are different for each class, and their values are higher than those of the first layer (left panel). These different outputs, after being processed by the final output classifier, result in distinguished outcomes for classification. The high standard deviation of $x(15)$ represents the sum over many transients that came from different initial conditions (samples from the same class).

In a different experiment, we investigated the behavior of a similar Resnet, with 15 layers, but with variable weight matrix for each layer. The mean and standard deviations of the residual for 200 samples are illustrated in figure 3, bottom panel. In this case, the residuals converge to their steady state solutions already on the fourth layer, as their standard deviations across samples converge to zero after the fourth layer. A striking finding in this case is that the standard deviation of the residuals for samples from different classes are zero, meaning that only one stable fixed point encodes all the similarities between different input classes. Considering this fact, we conclude that the sum of transient dynamics across layers for different input classes converges to different outputs that discriminate the inputs. Another interesting observation is that at the few layers close to the output layer, the weight matrix between layers converged to a fixed matrix. Also, compared to the previous example of a network with weight sharing, there are more residuals that converge to nonzero values. This gives the network enough capacity to give divergent outputs for different classes, based on their initial conditions. In this example, the classification error on the test set was about 1.8%. This higher value of the error rate might be due the paucity of separate fixed points to represent each class.

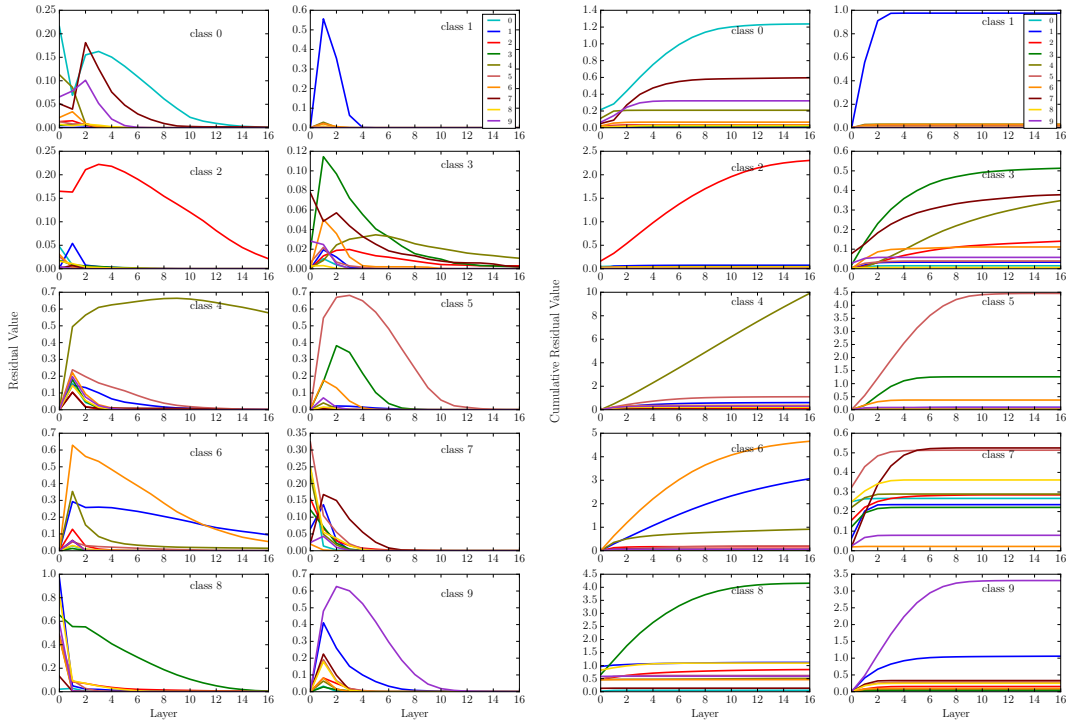

Figure 4: Residuals (left) and their cumulative (right) for neurons that have the largest contribution in the classifier's output for the softmax layer. In all cases, but class 8, the cumulative with the highest final value at layer 15 corresponds to the class that has the highest sensitivity to that neuron.

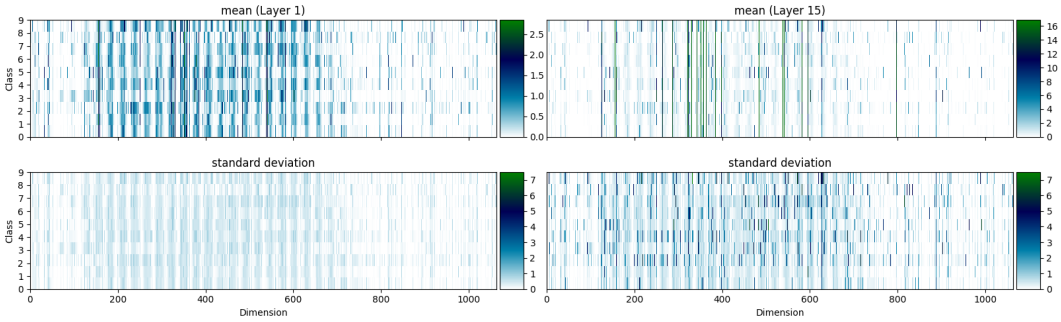

Figure 5: Mean and standard deviation of $x(t)$ at layer 1 (left panels), and layer 15 (right panels) for all ten classes of the input. The activities of neurons at layer 15 are different for different classes.

Considering the observation that a Resnet with multiple fixed points for the residuals, corresponding to different classes, renders a smaller classification error, hints to the point that having different similarity representations of the input encoded in the residuals results in a better generalization, compared to cases where only one single fixed point for residuals stands for the entire input classes.

5 ADAPTIVE DEPTH FOR RESNETS DURING TRAINING

Results of the previous sections shed some light on the mechanism of classification in Resnets. After understanding the role of transient dynamics in input classification, we envisage a new method to design the depth of Resnets based on the layer-dependent behavior of the residuals. In this method, during training, initially an arbitrary number of layers is chosen. After training each epoch using the back-propagation algorithm, the difference between the residuals for the last successive layers of the Resnet block are calculated. If this difference is less than a minimum threshold (we chose 0.01 for each neuron on average), the last hidden layer in the block is to be removed, because the value of the residuals will not contribute much to the cumulative function. This process continues until the network is trained (minimum loss on the training set). Note that convergence of the residuals is not a necessary requirement for classification, but a sufficient condition; i.e. when the residuals converge, and when the training loss is minimum, there is no need for extra layers in the blocks (and also before the classification layer). This algorithm can be implemented as a piece of code in parallel with other training algorithms for Resnets:

```
while loss function is not minimum do
    for each epoch of the training data do
        residuals of the last hidden layer in the block → r_1
        residuals of the second last hidden layer in the block → r_2
        calculate the l_1 norm for r_1 − r_2
        if l_1 norm < threshold then
            remove the layer corresponding to r_1
        end
    end
end
```

In the examples shown in the previous sections, we demonstrated a converging behavior of the residuals to stable or metastable (saddle fixed points) states. We applied this algorithm on the small network example with inputs from concentric circles, with a shared weight matrix between layers. We observed that a network with 5 hidden layers was also able to classify the test data with 100% accuracy. The same network with variable weights between layers required the maximum number of layers defined initially for training (15 layers).

6 CONCLUSION

In this study, we showed that given an input, Resnets integrate samples of the residuals from each layer, and build an output representation for the input data in the final hidden layer. This sum depends on the initial condition (input data) and its transition towards the steady state of the corresponding residual. In some networks which show attracting and converging behavior, one or more stable fixed point for the residuals exists. In other cases, among many other possible dynamics, multiple fixed points for different input classes might exist, some of which could be stable or metastable. In both cases, different neural transient dynamics (with inputs of different classes as initial conditions) can result in different cumulative values of the residuals, and therefore, different classification outcomes. We also developed a new method for designing an adaptive depth for Resnets during training. The main idea is that after all the residuals have settled into their steady state value, or if there are negligible changes of the values of the residuals between successive layers, there is no need for any extra deeper layers. This is because any additional layer of the residual neurons will add almost the same values as the previous layer, without any extra information about the neural transitions.

REFERENCES

Dean V Buonomano and Wolfgang Maass. State-dependent computations: spatiotemporal processing in cortical networks. *Nature reviews. Neuroscience*, 10(2):113–25, feb 2009. ISSN 1471-0048. doi: 10.1038/nrn2558. URL http://www.ncbi.nlm.nih.gov/pubmed/19145235.

Bo Chang, Lili Meng, Eldad Haber, Frederick Tung, and David Begert. Multi-level Residual Networks From Dynamical Systems View. *ICLR*, pp. 1–14, 2018.

Pratik Chaudhari, Adam Oberman, Stanley Osher, Stefano Soatto, and Guillaume Carlier. Deep Relaxation: partial differential equations for optimizing deep neural networks. *Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, PMLR*, 2017. URL http://arxiv.org/abs/1704.04932.

Brian Chu, Daylen Yang, and Ravi Tadinada. Visualizing Residual Networks.

Marco Ciccone, Marco Gallieri, Jonathan Masci, Christian Osendorfer, and Faustino Gomez. NAIS-NET: Stable Deep Networks from Non-Autonomous Differential Equations. *arXiv*, 2018.

Klaus Greff, Rupesh K. Srivastava, and Jürgen Schmidhuber. Highway and Residual Networks learn Unrolled Iterative Estimation. *ICLR*, (2015):1–14, 2017. URL http://arxiv.org/abs/1612.07771.

Eldad Haber and Lars Ruthotto. Stable Architectures for Deep Neural Networks. *arXiv*, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016. ISSN 1664-1078. doi: 10.1109/CVPR.2016.90. URL http://ieeexplore.ieee.org/document/7780459/.

Yiping Lu, Aoxiao Zhong, Quanzheng Li, Massachusetts General Hospital, and Bin Dong. Beyond Finite Layer Neural Networks: Bridging Deep Architectures and Numerical Differential Equations. *Arxiv*, pp. 1–15, 2017.

W Maass, T Natschlager, and Henry Markram. Real-time computing without stable states: a new framework for neural computation based on perturbations. *Neural {C}omput*, 14(11):2531–2560, 2002. doi: 10.1162/089976602760407955.

Ofer Mazor and Gilles Laurent. Transient dynamics versus fixed points in odor representations by locust antennal lobe projection neurons. *Neuron*, 48(November 23):661–673, 2005. ISSN 08966273. doi: 10.1016/j.neuron.2005.09.032.

Cbmm Memo No and Qianli Liao. Bridging the Gaps Between Residual Learning, Recurrent Neural Networks and Visual Cortex. *arXiv*, (047):1–16, 2016.

M. Rabinovich, R. Huerta, and G. Laurent. Transient dynamics for neural processing. *Science*, 321(July):48–50, 2008. ISSN 0036-8075. doi: 10.1126/science.1155564. URL http://www.sciencemag.org/cgi/content/full/321/5885/48{%}5Cnpapers://78a99879-71e7-4c85-9127-d29c2b4b416b/Paper/p14359{%}5Cnhttp://www.sciencemag.org/content/321/5885/48.short{%}5Cnhttp://cat.inist.fr/?aModele=afficheN{&}cpsidt=20493029.

Lars Ruthotto, Eldad Haber, and Computer Science. Deep Neural Networks motivated by Partial Differential Equations. *arXiv*, pp. 1–7, 2018.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training Very Deep Networks. *NIPS*, pp. 1–9, 2015.

Andreas Veit, Michael Wilber, Serge Belongie, and Cornell Tech. Residual Networks Behave Like Ensembles of Relatively Shallow Networks. *NIPS*, pp. 1–9, 2016.

Sergey Zagoruyko and Nikos Komodakis. Wide Residual Networks. *arXiv*, 2016.

7 APPENDIX

7.1 SENSITIVITY ANALYSIS OF THE OUTPUT WITH RESPECT TO LAYER PERTURBATIONS

As mentioned in the Introduction, some lesion studies have shown that weight perturbations at the initial layers of the network can have more sever consequences on output classification results than perturbations of the weights at deeper layers. Since we have considered a network with shared weights,we study the effect of perturbations of the residuals (which could be considered as the result of weight perturbations in previous studies). To understand how slight perturbations of the residuals affects the values of the output, we analyzed the sensitivity of the output C with respect to slight perturbations of the residuals $\mathbf{y}(t)$. Assuming slight perturbations on $\mathbf{y}(s)$, we are interested in the evolution of this perturbation throughout the network and its effect on the output unit. We used the chain rule of differentiation on the discrete time dynamics of the network to obtain a sensitivity matrix that propagates the perturbations from $\mathbf{y}(t)$, layer to layer, until it reaches the output $\mathbf{C}$ (the analysis is in discrete case). The sensitivity equation reads

$$S(t) = \frac{\partial \mathbf{C}}{\partial \mathbf{y}(s)} = \frac{\partial \mathbf{C}}{\partial \mathbf{x}(t)} \frac{\partial \mathbf{x}(t)}{\partial \mathbf{x}(t-1)} \frac{\partial \mathbf{x}(t-1)}{\partial \mathbf{x}(t-2)} \cdots \frac{\partial \mathbf{x}(s)}{\partial \mathbf{y}(s)} \tag{4}$$

where $t = T$ is the last hidden layer. Based on the definition of $\mathbf{x}(t)$, it is easy to verify that $\frac{\partial \mathbf{x}(t)}{\partial \mathbf{x}(t-1)} = \mathbf{I} + \frac{\partial \mathbf{y}(t)}{\partial \mathbf{x}(t-1)}$. We represent $\frac{\partial \mathbf{y}(t)}{\partial \mathbf{x}(t-1)}$ by $M(t)$, which is $\frac{\partial \mathbf{f}(W\mathbf{x}(t-1))}{\partial \mathbf{x}(t-1)}$. Using vector representations, $M = \mathbf{y}(t) \odot (1 - \mathbf{y}(t)) \odot W$, where $\odot$ represents element-wise multiplications. Since the output class $\mathbf{C}$ is the result of the inner product between the classifier vector K and the cumulative signal $\mathbf{x}(T)$, $\frac{\partial \mathbf{C}}{\partial \mathbf{x}(T)} = K$. It is also clear that the last term on the right hand side of equation 4 is equal to the identity matrix. Therefore, in equation 4, the sensitivity matrix can be represented as

$$S(t) = K[I + M(T)][I + M(T-1)][I + M(T-2)] \cdots [I + M(s)] = KM^{'}(s) \tag{5}$$

For different layers s_1 and s_2, where $s_2 > s_1$, we calculated $M^{'}(s_1)$ and $M^{'}(s_2)$. It turns out that for network simulations that we performed (see Experiment section), for $s_2 = 10$ and $s_1 = 4$, the determinant of $M^{'}(s_1)$ is orders of magnitude larger than the determinant of $M^{'}(s_2)$. This implies that perturbations at initial layers are greatly amplified at the final hidden layer compared to perturbations at close to final layers. Since the final hidden layer is multiplied by the classifier vector K, those amplified perturbations will have a more disruptive consequence on the output of the network.

7.2 LONG-TIME BEHAVIOR OF THE RESIDUALS ON MNIST DATASET

Initially, we trained a Resnet with 15 layers with shared weights between layers. To observe the long-time behavior of the residuals in this network, we used the same weights in a network with 1000 hidden layers (without retraining). There were a few non-zero fixed points with some negligible standard deviation among 200 samples. To check if the fixed points were different and distinguished for each class, we plotted the average and the standard deviation of the residuals for the test set, separately for each class, on the final hidden layer in figure 6. The average for each class is different from any other class, however, a large number of dimensions are identical. The fact that each class has a distinguished fixed point indicates that the trajectories of the residuals for each class are separated. According to the bottom panel of figure 6, the standard deviation between the residuals in a single class are small, and negligible, and indicate that the classification accuracy is not 100%.

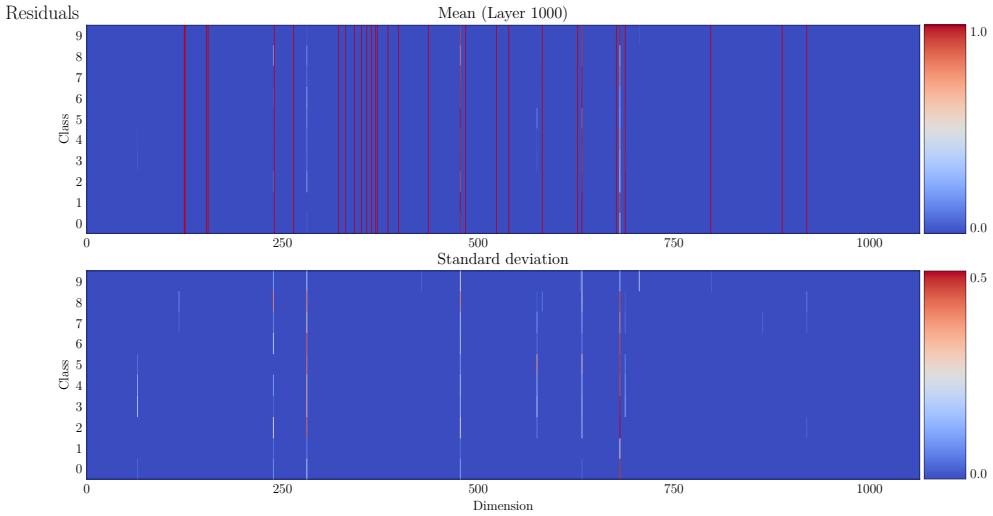

Figure 6: Mean (top) and standard deviation (bottom) of the residuals, for the network with shared weights, corresponding to the 10 different classes of the MNIST test data. The mean of the residuals are different in few dimensions for each class.