Neural Data Augmentation: Interim Report

Group 66 – S1740192, S1778089

Abstract

Deep learning is currently the driving force in a wide range of applications. However, deep neural networks often require a large amount of data. Traditional approaches mitigate the problem of small training sets by augmenting the data using a series of known transformations. Recent advances in unsupervised learning have proposed the use of neural networks as powerful black-box generative models. We aim to investigate to which extent data augmentation based on neural generative models can be an effective strategy and whether combining these models with traditional approaches can further improve classification performance. For this purpose, we start by evaluating the behavior of deep neural network classifiers on image recognition for training sets of varying size. In particular, we evaluate the performance of two classifiers, VGG-Net and CapsNet, on data sets that have been built based on MNIST and CIFAR10. We observe that as the training set becomes smaller, the accuracy decreases, especially on CIFAR10 which poses a more difficult image recognition task.

1. Introduction

The field of deep learning has gained a massive amount of attention among machine learning researchers and practitioners alike (LeCun et al., 2015). Indeed, deep neural networks seem to excel at perception tasks and, as a result, have revolutionized several disciplines (Schmidhuber, 2015), ranging from computer vision and natural language processing to speech recognition and time series analysis. However, in order to achieve such state-of-art performances, their architectures are typically complex and a recent trend is to consider increasingly deeper designs (Gu et al., 2017). This naturally poses a number of challenges as not only the training procedure becomes computationally difficult and expensive, but also requires the collection of large-scale data sets. Traditional collection of labeled data sets is thus rapidly becoming impracticable because it often involves human intervention. Some researchers have adopted crowdsourcing services that allow large-scale annotation (Russakovsky et al., 2015), but this approach raises new problems and does not remove humans from the loop. Others have considered the use of external data from online search engines (Xie et al., 2014) or even transfer learning schemes (Yosinski et al., 2014), where for instance the network is trained on a large-scale data set with a possibly different task and then fine-tuned on the (small-scale) target data set. In the small data setting, Bayesian neural networks have also shown promising results (Gal & Ghahramani, 2015).

An alternative approach is to consider data augmentation, where the data set is artificially enlarged by new synthesized observations. For this purpose, traditional methods use a combination of known transformations, which for image data includes rotations, translations and additive noise (Krizhevsky et al., 2012); whereas for audio it involves for instance time stretching and pitch shifting (Salamon & Bello, 2017). Many of these augmentation schemes have been shown to be of critical importance in deep learning, but are clearly sub-optimal in the sense that they often require human intervention and possibly extensive domain knowledge. Therefore, these schemes constitute feature engineering, albeit in a different guise. Not surprisingly, recent advances posit that data augmentation strategies can be automatically learned. In (Hauberg et al., 2016), the authors propose the random generation of diffeomorphisms on a per-class basis by learning the parameters of class-specific generative models and proceed to show the benefits of this approach in terms of classification performance when compared to traditional schemes. On the other hand, inspired by the development of deep (neural) generative models, such as Generative Adversarial Network (GAN) (Goodfellow et al., 2014), the authors in (Antoniou et al., 2017) propose a GAN variant that is again able to learn plausible transformations and can generalize such transformations to unseen classes since the generation process is itself class-agnostic.

In this work, we also draw inspiration from neural generative models, specifically GAN and Variational Autoencoder (VAE) (Kingma & Welling, 2013), in order to generate synthetic observations from the data. This approach will henceforth be referred as neural data augmentation. Indeed, we believe that these black-box generative models contain enough representational power as to learn plausible transformations while, in principle, requiring minimal assumptions from the practitioner regarding the data. However, since the procedure is not limited to learning the parameters of the generative distribution, but also its implicit and explicit functional form (GAN and VAE respectively), their applicability might be limited, especially so when the training data size is exceedingly small. As a result, our work is most related to (Antoniou et al., 2017), but we do not explore the same models and challenges. In particular, we are interested in investigating to which extent data augmentation using GANs and VAEs can be an effective strategy and whether combining this approach with traditional schemes can be beneficial. The rest of this report is organized as follows: in Section 2, we discuss our objectives in more detail and, in Section 3, we describe the task and data sets to be explored. Section 4 provides a description of the methods used to generate our set of baseline results and these experiments are presented in Section 5. Finally, we provide interim conclusions in Section 6 and point to future plans in Section 7.

2. Objectives

As previously mentioned, our primary aim is to explore under which circumstances neural data augmentation is suitable. Thus, we are interested in measuring how classification performance varies under different data regimes and whether neural data augmentation can improve these performances. For this purpose, we start by evaluating the behavior of two distinct deep neural network classifiers on increasingly smaller data sets without data augmentation. One is a convolutional neural network (CNN) whose architecture is based on homogeneous convolutional layers and has been proven to yield good results (Simonyan & Zisserman, 2014). The other classifier is a relatively recent development that tries to address some of the inherent flaws in CNNs by extending them with capsule layers (Sabour et al., 2017). The authors claim that the resulting architecture is better able to represent data and should in principle require less training observations. Since we are particularly interested in the small data regime, this development can perhaps be key in the design of more efficient models, including neural generative models.

In a future phase, we delve into data augmentation strategies, aiming to address our primary research question by comparing the results with those obtained during the first phase. We consider neural data augmentation based on both conditional GAN and VAE (Mirza & Osindero, 2014; Sohn et al., 2015) and we will analyze these models in terms of robustness to different data regimes (consistency) as well as their ability to improve classification performance. Similarly, we might investigate the relationship between classification performance (accuracy) and the quality of the synthetic data, measured for instance by the Inception score metric (Salimans et al., 2016) or the Fréchet Inception distance (Heusel et al., 2017) – we are aware of their limitations, but provably better alternatives seem to be lacking (Barratt & Sharma, 2018). In addition, a particularly important aspect is the comparison between traditional and neural data augmentation and whether combining the two approaches can further boost accuracy. A thorough analysis might be difficult to accomplish due to time and computational constraints, but we believe that a minimal setup is feasible. Finally, a completely optional objective is related to the improvement of neural generative models, namely conditional GAN and VAE. For instance, we might investigate whether architectures based on capsules help improve the quality of the data being generated and further analyze their consequence in terms of classification performance.

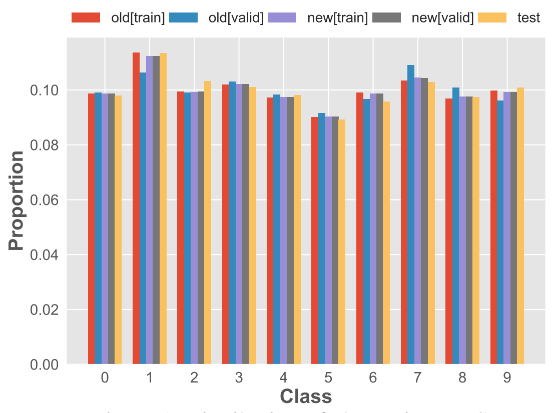

Figure 1. Distribution of classes in MNIST.

3. Data set and task

To evaluate whether neural data augmentation is beneficial, we focus on the task of image recognition. We explore two data sets of different complexity, but with the same number of classes (10): MNIST (LeCun et al., 1998) and CIFAR10 (Krizhevsky & Hinton, 2009). In MNIST, each sample is a 28×28 pixel gray-scale image of a handwritten digit; whereas, in CIFAR10 each observation corresponds to a 32×32 pixel color image of vehicles (airplane, automobile, ship, truck) and animals (bird, cat, deer, dog, frog, horse). Both data sets contain 10,000 images as test data, but while the number of classes in the test set of CIFAR10 is completely balanced, on MNIST it is only approximately so. Additionally, in an attempt to balance the number of classes in the training/validation sets and make MNIST and CIFAR10 comparable with respect to the size of the training set, we have considered a split that is different from the one that has been provided. In particular, for each data set, we have considered a stratified sampling approach which generates training sets of size 42,500, using the remainder observations for validation purposes (17,500 and 7,500 for MNIST and CIFAR10 respectively). Using this procedure, we were able to achieve a perfect class balance in CIFAR10, but not in MNIST. However, compared to the previous split, this constitutes an improvement (see Figure 1).

More importantly, since we are interested in the evaluation of classifiers (accuracy) under different data regimes, we consider training subsets of different sizes. In particular, we consider versions with 90%, 80%, ..., 10%, 5%, 2% and 1% of the full training set (100%) – we henceforth refer to each of these as MNIST-α (CIFAR10-α), where α denotes a given relative percentage. These versions are generated by successive stratified sampling. For instance, MNIST-90 is a subset of MNIST-100, MNIST-80 is a subset of MNIST-90, and so forth. Note that ideally experiments should be performed over multiple random splits so that the reported results are not influenced by a specific set of splits, but due to computational constraints we are unable to follow this route. Finally, at some point in our experiments, we have also considered a cropped version of CIFAR10, where each 32×32 image has been randomly cropped to 24×24. This modified version is referred as RCCIFAR10.

4. Methodology

In this section, we provide a specification of the two classifiers. However, due to space constraints, we are unable to provide descriptions and analyses as comprehensive as we would like. Instead, we point the reader to relevant literature for complementary information. For a similar reason, we do not describe generative models. These models will be explained in our final report.

4.1. VGG-Net

CNNs have been highly successful (Schmidhuber, 2015) and unsurprisingly there has been a constant effort to build upon the architecture suggested in (Krizhevsky et al., 2012). Currently, the best performing architectures consider more advanced layer designs, including the use of different filter sizes to capture visual patterns at multiple scales as well as shortcut connections that allow effective training of deeper networks (Gu et al., 2017). Historically, a major development was proposed by (Simonyan & Zisserman, 2014), where state-of-art results were achieved using a homogeneous architecture containing small-sized filters (VGG-Net). This aspect revealed to be particularly effective as it allowed the incorporation of more non-linearities, while reducing the number of weights, when compared to networks with larger receptive fields.

In this work, due to several constraints, we examine simpler variants of the VGG-Net. The architectures have 3 convolutional stages with 64, 128 and 256 filters respectively. Each block consists of two convolutional layers with 3×3 kernels and weights are initialized with the Glorot Uniform scheme (Glorot & Bengio, 2010). We only consider the use of Leaky ReLU because, in the previous coursework, we found that this activation function can lead to good performing CNNs – a more detailed evaluation is presented in (Xu et al., 2015). After each convolutional operation, we optionally apply batch normalization (Ioffe & Szegedy, 2015). Dimensionality reduction at each stage is carried out using convolutions with stride 2 or, alternatively, max-pooling (2×2, stride 2), in which case convolutions with stride 1 are used instead. After each block, we allow the possibility of dropout (Srivastava et al., 2014). Finally, a single fully-connected layer with softmax is added.

4.2. Capsule Network

It is postulated that effective computer vision systems must be able to translate knowledge across space. Typical CNNs achieve this by relying on shared feature detectors and max-pooling. The latter operation is known to provide translational invariance, albeit in a rudimentary fashion. Indeed, CNNs seem to be inherently flawed as they struggle at modeling objects from different viewpoints due to lack of information regarding pose, requiring in turn a significant amount of data. In (Sabour et al., 2017), the authors investigate capsule layers which try to overcome these shortcomings by directly encoding viewpoint invariance in transformation matrices. In addition, a dynamic routing between capsule layers is also proposed, where capsules at a lower layer actively select the capsule in the layer above that is better able to handle such information. The authors believe this leads to a more faithful characterization of the human vision system, constituting a more effective approach.

Layer		Shape MNIST	Shape CIFAR10
	Image	$28 \times 28 \times 1$	$32 \times 32 \times 3$
ReLU Conv1	Conv. layer with 9×9 kernel, 256 channels, stride 1 and no padding	$20 \times 20 \times 256$	$24 \times 24 \times 256$
PrimaryCaps	Conv. capsule layer with 32 8D capsules, 9×9 kernel, stride 2 and no padding	$32 \times 6 \times 6 \times 8$	$32 \times 8 \times 8 \times 8$
DigitCaps	Fully-connected capsule layer with 16D capsule per class (output vector $\boldsymbol{v}_j$)	10×16	10×16
ReLU FC1	Fully-connected with ReLU	512	512
ReLU FC2	Fully-connected with ReLU	1024	1024
Sigmoid FC	Fully-connected with sigmoid	784	3072

Table 1. Capsule network (CapsNet) architecture. The second section corresponds to the structure of the optional decoder, a fully-connected network that reconstructs an image from the DigitCaps representation.

A capsule corresponds to a group of neurons that captures not only the likelihood, but also the various properties of a specific entity in an image. In particular, each component of the output vector $\boldsymbol{v}_j$ of capsule j encodes a given property, while its magnitude yields the existence probability of such entity. The non-linearity applied to the its total input vector $\boldsymbol{s}_j$ ensures that the magnitude of $\boldsymbol{v}_j$ is between 0 and 1:

$$\boldsymbol{v}_j = \frac{||\boldsymbol{s}_j||^2}{1 + ||\boldsymbol{s}_j||^2} \frac{\boldsymbol{s}_j}{||\boldsymbol{s}_j||}. \tag{1}$$

For each capsule j, the output $\boldsymbol{u}_i$ of capsule i in the previous layer is transformed using the transformation matrix $\boldsymbol{W}_{ij}$ to produce the prediction vector $\hat{\boldsymbol{u}}_{j|i}$. Excluding the first capsule layer, the total input $\boldsymbol{s}_j$ is given by a weighted sum of the prediction vectors $\hat{\boldsymbol{u}}_{j|i}$:

$$\boldsymbol{s}_j = \sum_i c_{ij}\, \hat{\boldsymbol{u}}_{j|i}, \qquad \hat{\boldsymbol{u}}_{j|i} = \boldsymbol{W}_{ij}\, \boldsymbol{u}_i, \tag{2}$$

where c_{ij} are the coefficients determined by the dynamic routing procedure. First, a similarity score b_{ij} is computed, consisting of a sum of a prior coupling probability with the scalar product between the prediction vector $\hat{\boldsymbol{u}}_{j|i}$ and the resulting output vector $\boldsymbol{v}_j$. The vector of coupling coefficients $\boldsymbol{c}_i$ is then found by the application of softmax to vector $\boldsymbol{b}_i$ and subsequently refined – the authors suggest 3 iterations. Refer to (Sabour et al., 2017) for additional details and (Hinton et al., 2018) for an alternative routing process.

Furthermore, in order to train the network weights, the authors propose a new loss function. In particular, a separate margin loss L_c for each top-level capsule c, also referred as DigitCaps (the authors perform digit identification on MNIST), is devised:

$$\begin{aligned} L_c = {} & T_c \max(0,\ m^+ - ||\boldsymbol{v}_c||)^2 + \\ & \lambda\,(1 - T_c) \max(0,\ ||\boldsymbol{v}_c|| - m^-)^2, \end{aligned} \tag{3}$$

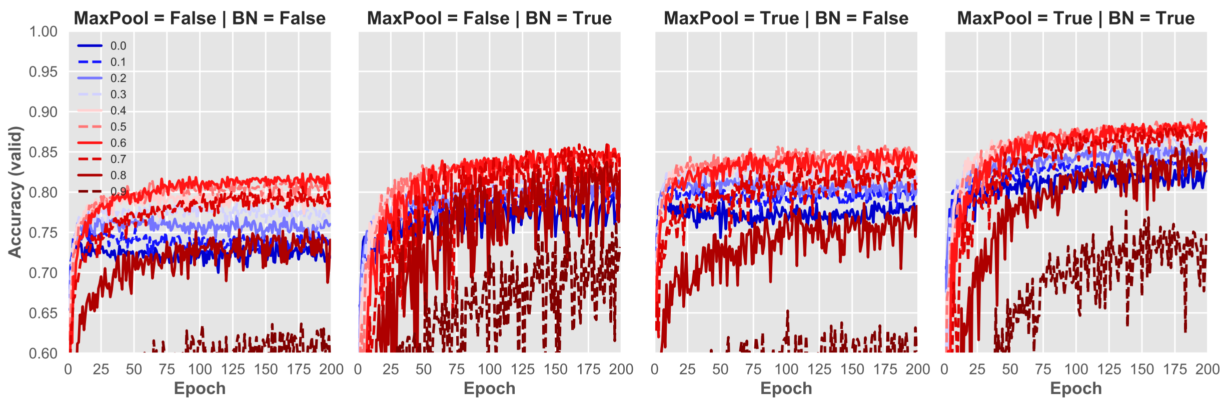

Figure 2. Accuracy on CIFAR10 validation set using different VGG-Net configurations. Full training set is used: CIFAR10-100.

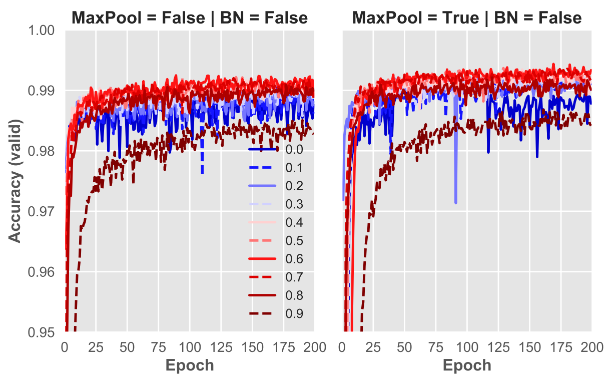

Figure 3. Accuracy on MNIST validation set using different VGG-Net configurations. Full training set is used: MNIST-100. Batch normalization led to unstable behavior during training.

where $T_c = 1$ if class c is present (0 otherwise), $m^+ = 0.9$, $m^- = 0.1$ and λ (default 0.5) tries to avoid scaling down the activity (output) vectors during initial learning. The total loss is the sum of all L_c. A reconstruction loss, measuring the discrepancy in terms of sum of square differences between the reconstructed and original images, is also added to the total loss with a scaling factor of 0.0005. This enables the network to capture other critical properties of the image.

Regarding the capsule network (CapsNet) architecture, we follow the same as proposed in (Sabour et al., 2017), where a single convolutional layer precedes two capsule layers (PrimaryCaps and DigitCaps) and the decoder takes the outputs from the DigitCaps layer and applies 2 fully-connected layers with ReLU activations, followed by a single fully-connected layer with sigmoid activation. This is summarized in Table 1. Note that several implementations available online perform classification on the simpler MNIST data set. For our experiments, we adopt a TensorFlow implementation for MNIST (Liao, 2018) and extend it to a more general case, including the ability to handle images from CIFAR10. Several other modifications are required so as to fit within our experimentation framework. Finally, it is also worth pointing that an alternative implementation is provided by one of the authors in (Sabour, 2018). This version considers the use of an ensemble for classification on CIFAR10, which naturally leads to better performance, but is computationally more expensive.

5. Experiments

Before proceeding with the analysis of our results, we note that our networks are optimized using Adam with default parameters (Kingma & Ba, 2014). This choice is mainly due to its popularity, since it has been recently shown that it can lead to abnormal behavior due in part to exponential moving averages. This has led the authors in (Reddi et al., 2018) to propose AMSGrad. Note that abnormal behavior has also been observed in our previous work. However, stochastic optimization methods are not the focus of this work and, as such, alternative approaches are unlikely to be tested due to computational constraints.

In the first set of experiments, we determine which VGG-Net configuration produces good results in terms of accuracy. For this purpose, we could employ a Bayesian optimization scheme (Snoek et al., 2012), but we decide not to follow this route because our primary focus is not in obtaining the best classification performance possible – again, we are interested in the relative gain that neural data augmentation might yield. Hence, we train multiple configurations of VGG-Net for 200 epochs, restricting the configuration space to different dropout rates (0 to 0.9 with steps of 0.1), the possibility of having batch normalization and a dimensionality reduction method that, as described in Section 4, is either based on convolutions with stride 2 or max-pooling. Figures 2 and 3 show the performance curves of the different configurations on CIFAR10 and MNIST respectively. Note that, for MNIST, the curves corresponding to batch normalization have not been included because in our experiments this led to unstable behavior during training, specifically this behavior became more noticeable after training for 50 epochs. For both data sets, we observe that a dropout rate of 0.6 consistently leads to one of the best performances, regardless of which dimensionality reduction method is being used or whether batch normalization is enabled (CIFAR10). Max-pooling also reveals to be advantageous for both data sets. Given our restricted configuration space, the optimal VGG-Net for CIFAR10 has a dropout rate of 0.6, max-pooling and batch normalization, with the best validation accuracy being 88.8% (epoch 195). The optimal VGG-Net for MNIST has a similar configuration (batch normalization disabled), achieving a validation accuracy of 99.4% (epoch 169). It is also worth mentioning

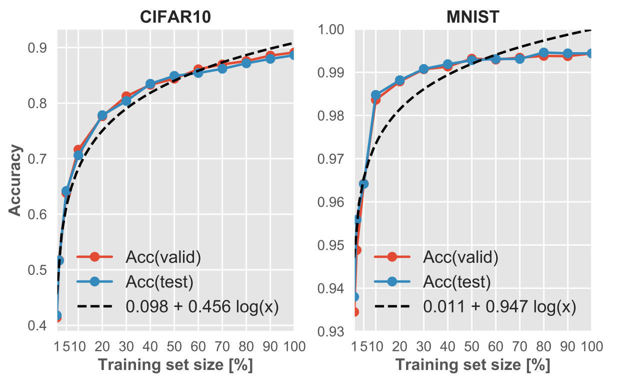

Figure 4. Validation and test accuracy using optimal VGG-Nets on CIFAR10 and MNIST with varying training set sizes (learning curves). Logarithmic curves have been fitted to test accuracy.

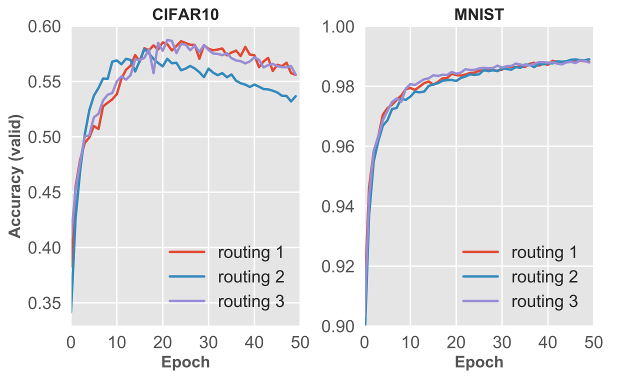

Figure 5. Validation accuracy on CIFAR10 and MNIST using CapsNet with different number of routing iterations (full training sets).

that the higher accuracy and faster convergence on MNIST is due to the complexity of the task, which is significantly easier when compared to CIFAR10.

Having determined the optimal VGG-Net configuration for both data sets, the next set of experiments involved training the classifier on subsets of the full training data for 200 epochs. As described in Section 3, for CIFAR10, this process corresponds to training on all subsets from CIFAR10-90 to CIFAR10-1, operating in similar fashion for MNIST. Due to space constraints we are unable to show the corresponding performance curves. Instead, Figure 4 provides a summary of these experiments. In particular, for each subset, we register the best observed validation accuracy and its corresponding test accuracy, yielding a simplified learning curve. There are at least two noteworthy observations. First, the classifier exhibits consistent behavior since, for all subsets, the validation accuracy and the resulting test accuracy are not too dissimilar: they do not differ by more than 1% on CIFAR10 and 0.7% on MNIST. Second, both curves appear to follow approximately a logarithmic trendline, especially so for CIFAR10 where we observe a R^2 of 0.980 compared to 0.911 for MNIST. Surprisingly, the test accuracy of the classifier on MNIST is always higher than 93.8%, unlike CIFAR10 where the worse performance is 41.8%. This is again an evidence that the task posed by MNIST is significantly easier than that of CIFAR10. In both data sets, we observe a sharper decrease in performance for training set sizes below 20%.

In the next set of experiments, we turn our attention to the

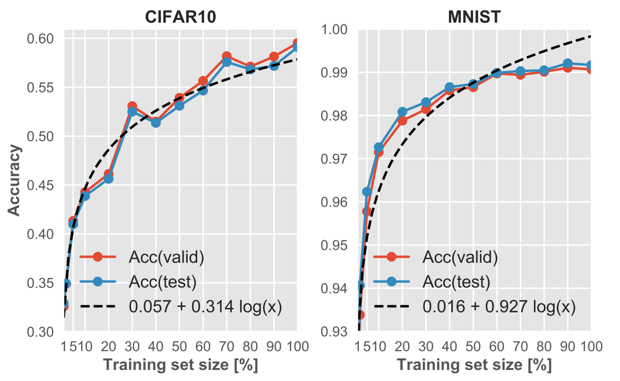

Figure 6. Validation and test accuracy using CapsNet (routing 1) on CIFAR10 and MNIST with varying training set sizes (learning curves). Logarithmic curves have been fitted to test accuracy.

capsule network (CapsNet) classifier. In particular, we start by testing its behavior with respect to the number of routing iterations. For this purpose, we train the classifier with its default architecture on CIFAR10 and MNIST, varying the routing iterations from 1 to 3. From Figure 5, we observe that the increase in computational complexity due to a higher number of routing iterations does not seem to be advantageous. For instance, on CIFAR10, which is the data set that has yield the most dissimilar performance curves, the best validation accuracy does indeed occur when the number is 3 (58.7%, epoch 21), but the difference in performance compared to 1 routing iteration (58.6%, epoch 24) is negligible. The worse performance occurs when this number is 2, but it would be unwise to conclude that this holds in general based only on this set of experiments. Fundamentally, in order to extract any statistically significant result more experimentation would be required, which given the time constraints was not possible. A particular important observation is not however related to the number of iterations, but instead to the apparently bad performance of CapsNet on the CIFAR10 data set. In an attempt to improve its performance, we set a number of pilot experiments where we consider different regularization coefficients for the reconstruction loss and, as suggested in (Sabour et al., 2017), a cropped version of the data set (RCCIFAR10). Nevertheless, none of these modifications seem to boost the performance to a level that is comparable to that of VGG-Net. We also briefly consider increasing the number of capsules and their dimensions, but this quickly leads to models that either take too long to train or are too large. For instance, doubling these numbers yields a model whose parameters do not seem to fit in 3 GPUs. It might however be possible to write code optimized for multiple GPUs so as to avoid these problems. Given the limited time and resources, we have not been able to test them reliably. As a result, we reluctantly adopt the default architecture with 1 routing iteration in the subsequent experiments. For complementary studies on the empirical evaluation of CapsNet, we refer the reader to (Sabour et al., 2017; Xi et al., 2017).

In our final set of experiments, we proceed similarly as when testing the VGG-Net. In particular, we train the CapsNet classifier on subsets of the full training data for 200 epochs. Again, due to space constraints we only provide a summary of these experiments: for each subset, we register the best observed validation accuracy and its corresponding test accuracy. Figure 6 depicts the learning curves of CapsNet on both CIFAR10 and MNIST. For consistency purposes, we fit logarithmic curves to the empirical curve (test accuracy), obtaining a R^2 of 0.977 and 0.919 on CIFAR10 and MNIST respectively. We observe again that as the training set becomes smaller, the accuracy decreases and that this decrease is sharper for training set sizes below 20%. This decline is also more evident on CIFAR10 because it poses a more difficult image recognition task. Perhaps surprisingly, CapsNet exhibits the same consistent behavior as VGG-Net, with validation and test accuracy not differing by more than 1% on CIFAR10 and 0.7% on MNIST. However, the performance of this particular CapsNet architecture appears to be worse than VGG-Net on both data sets. In particular, while VGG-Net achieves a maximum test accuracy of 88.6% and 99.5% on CIFAR10 and MNIST, CapsNet tops at around 59.0% and 99.2% respectively. Admittedly, this difference on MNIST is negligible and in order to extract a statistically significant result more experimentation would be required. On the other hand, CapsNet appears to scale worse when varying the size of the training set, bottoming out at roughly 90.9% on MNIST compared to 93.8% using VGG-Net. Finally, this particular CapsNet is clearly not suitable for image recognition on CIFAR10, mainly due to the existence of diverse backgrounds and, as discussed in (Sabour et al., 2017), the need to explain all components in an image.

6. Interim conclusions

In this report, we started by introducing several approaches whose main objective is to mitigate the negative impact of training complex deep neural networks on comparatively small data sets, i.e. networks whose number of parameters exceeds the number of training observations. While some approaches consider the use of external data, transfer learning schemes and Bayesian versions of standard neural networks, in this project we will focus on data augmentation strategies. Traditional data augmentation relies on a series of known transformations, but it is hypothesized that these transformations can be learned from data. For this purpose, we consider black-box neural generative models, referring to this approach as neural data augmentation.

During this first phase, we have identified image recognition as being an appropriate task and have built base data sets with training sets of varying size from MNIST and CIFAR10. We have considered two neural network classifiers: one based on relatively mature ideas (VGG-Net) and another that combines these ideas with novel concepts (CapsNet). We then evaluated their behavior without adopting any data augmentation strategy – this constitutes our set of baseline results and it will be used to determine whether neural data augmentation can improve classification performance. As expected, we have observed that as the training set becomes smaller, the accuracy decreases, especially on CIFAR10 which poses a more difficult recognition task.

Furthermore, both classifiers perform similarly on MNIST, with VGG-Net being at a slight advantage. On CIFAR10 however CapsNet are markedly worse which is due to the existence of complex backgrounds and the need to explain all entities in an image. We hypothesize that devising a method that can mask the background before feeding images to CapsNet can significantly improve performance. Another possible approach is to increase the number of capsules and their dimensions, effectively boosting the expressive power of the network.

7. Plan

The first phase of our work mainly focused on finding optimal configurations for our baseline classifiers and evaluating their performance on several data sets by varying the corresponding training set sizes. This in turn allows us to simulate different data regimes. In the next phase, we expand this analysis by focusing on data augmentation as a strategy to improve classification performance. At this point, we would like to highlight that, given the superior performance of VGG-Net over the more recent CapsNet, we are unlikely to consider the latter as our primary classifier. However, capsule networks constitute an active research topic, and we have suggested some ideas that might yield promising results. Consequently, we might revisit these networks in the future, but only after completing our main objectives.

The next milestone is to have working implementations of generative models, particularly conditional GANs (Mirza & Osindero, 2014) and conditional VAEs (Kingma et al., 2014; Sohn et al., 2015). Conditional generative models will allow us to augment the data sets with labeled synthetic observations. In this context, we might also introduce modifications so as to improve their performance, using for instance the ideas suggested in (Makhzani et al., 2015; Salimans et al., 2016; Srivastava et al., 2017). After a preliminary evaluation, we plan to extend the code provided in (Kristiadi, 2018), adapting it to fit within our experimentation framework. Since these implementations focus on MNIST, additional work will involve tuning the models for CIFAR10. More importantly, we will setup a data pipeline that gathers the output from the generative models and feeds it to the classifiers as training data. Using this system and proceeding in a manner that is similar to that presented in this report, we are able to answer our primary research question by examining to which extent neural data augmentation is an effective strategy. In particular, we will analyze the generative models in terms of robustness to different data regimes as well as their ability to improve classification performance. At the same time, we will compare and extend this approach with traditional data augmentation strategies, using rotated, translated and noisy images. We are aware that a thorough investigation might involve a significant amount of work, but we believe that a minimal setup is certainly feasible. Optionally, we might also explore the relationship between visually meaningful images and classification performance.

References

Antoniou, Antreas, Storkey, Amos, and Edwards, Harrison. Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340*, 2017.

Barratt, Shane and Sharma, Rishi. A note on the inception score. *arXiv preprint arXiv:1801.01973*, 2018.

Gal, Yarin and Ghahramani, Zoubin. Bayesian convolutional neural networks with bernoulli approximate variational inference. *arXiv preprint arXiv:1506.02158*, 2015.

Glorot, Xavier and Bengio, Yoshua. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 249–256, 2010.

Goodfellow, Ian, Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron, and Bengio, Yoshua. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Gu, Jiuxiang, Wang, Zhenhua, Kuen, Jason, Ma, Lianyang, Shahroudy, Amir, Shuai, Bing, Liu, Ting, Wang, Xingxing, Wang, Gang, Cai, Jianfei, et al. Recent advances in convolutional neural networks. *Pattern Recognition*, 2017.

Hauberg, Søren, Freifeld, Oren, Larsen, Anders Boesen Lindbo, Fisher, John, and Hansen, Lars. Dreaming more data: Class-dependent distributions over diffeomorphisms for learned data augmentation. In *Artificial Intelligence and Statistics*, pp. 342–350, 2016.

Heusel, Martin, Ramsauer, Hubert, Unterthiner, Thomas, Nessler, Bernhard, and Hochreiter, Sepp. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pp. 6629–6640, 2017.

Hinton, Geoffrey E., Sabour, Sara, and Frosst, Nicholas. Matrix capsules with EM routing. *International Conference on Learning Representations (forthcoming)*, 2018.

Ioffe, Sergey and Szegedy, Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456, 2015.

Kingma, Diederik P and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, Diederik P and Welling, Max. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kingma, Diederik P, Mohamed, Shakir, Rezende, Danilo Jimenez, and Welling, Max. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pp. 3581–3589, 2014.

Kristiadi, Agustinus. generative-models: A collection of generative models in pytorch and tensorflow, 2018. URL `https://github.com/wiseodd/generative-models`.

Krizhevsky, Alex and Hinton, Geoffrey. Learning multiple layers of features from tiny images. 2009.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

LeCun, Yann, Bottou, Léon, Bengio, Yoshua, and Haffner, Patrick. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

LeCun, Yann, Bengio, Yoshua, and Hinton, Geoffrey. Deep learning. *Nature*, 521(7553):436, 2015.

Liao, Huadong. Capsnet-tensorflow: A tensorflow implementation of capsule networks, 2018. URL `https://github.com/naturomics/CapsNet-Tensorflow`.

Makhzani, Alireza, Shlens, Jonathon, Jaitly, Navdeep, Goodfellow, Ian, and Frey, Brendan. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

Mirza, Mehdi and Osindero, Simon. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

Reddi, Sashank J., Kale, Satyen, and Kumar, Sanjiv. On the convergence of adam and beyond. *International Conference on Learning Representations (forthcoming)*, 2018.

Russakovsky, Olga, Deng, Jia, Su, Hao, Krause, Jonathan, Satheesh, Sanjeev, Ma, Sean, Huang, Zhiheng, Karpathy, Andrej, Khosla, Aditya, Bernstein, Michael, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

Sabour, Sara. Sarasra: Tensorflow research models, 2018. URL `https://github.com/Sarasra/models/tree/master/research/`.

Sabour, Sara, Frosst, Nicholas, and Hinton, Geoffrey E. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems*, pp. 3859–3869, 2017.

Salamon, Justin and Bello, Juan Pablo. Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE Signal Processing Letters*, 24(3):279–283, 2017.

Salimans, Tim, Goodfellow, Ian, Zaremba, Wojciech, Cheung, Vicki, Radford, Alec, and Chen, Xi. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pp. 2234–2242, 2016.

Schmidhuber, Jürgen. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

Simonyan, Karen and Zisserman, Andrew. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Snoek, Jasper, Larochelle, Hugo, and Adams, Ryan P. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pp. 2951–2959, 2012.

Sohn, Kihyuk, Lee, Honglak, and Yan, Xinchen. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems*, pp. 3483–3491, 2015.

Srivastava, Akash, Valkoz, Lazar, Russell, Chris, Gutmann, Michael U, and Sutton, Charles. Veegan: Reducing mode collapse in gans using implicit variational learning. In *Advances in Neural Information Processing Systems*, pp. 3310–3320, 2017.

Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1): 1929–1958, 2014.

Xi, Edgar, Bing, Selina, and Jin, Yang. Capsule network performance on complex data. *arXiv preprint arXiv:1712.03480*, 2017.

Xie, Saining, Yang, Tianbao, Wang, Xiaoyu, and Lin, Yuanqing. Hyper-class augmented and regularized deep learning for fine-grained image classification. *CVPR2015*, 2014.

Xu, Bing, Wang, Naiyan, Chen, Tianqi, and Li, Mu. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

Yosinski, Jason, Clune, Jeff, Bengio, Yoshua, and Lipson, Hod. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pp. 3320–3328, 2014.