MultiplexNet: Towards Fully Satisfied Logical Constraints in Neural Networks

Nicholas Hoernle,[1] **Rafael Michael Karampatsis,** [1] **Vaishak Belle** [1, 2]**, Kobi Gal** [1, 3]

[1] University of Edinburgh
[2] Alan Turing Institutety
[3] Ben-Gurion University

Abstract

We propose a novel way to incorporate expert knowledge into the training of deep neural networks. Many approaches encode domain constraints directly into the network architecture, requiring non-trivial or domain-specific engineering. In contrast, our approach, called MultiplexNet, represents domain knowledge as a quantifier-free logical formula in disjunctive normal form (DNF) which is easy to encode and to elicit from human experts. It introduces a latent Categorical variable that learns to choose which constraint term optimizes the error function of the network and it compiles the constraints directly into the output of existing learning algorithms. We demonstrate the efficacy of this approach empirically on several classical deep learning tasks, such as density estimation and classification in both supervised and unsupervised settings where prior knowledge about the domains was expressed as logical constraints. Our results show that the MultiplexNet approach learned to approximate unknown distributions well, often requiring fewer data samples than the alternative approaches. In some cases, MultiplexNet finds better solutions than the baselines; or solutions that could not be achieved with the alternative approaches. Our contribution is in encoding domain knowledge in a way that facilitates inference. We specifically focus on quantifier-free logical formulae that are specified over the output domain of a network. We show that this approach is both efficient and general; and critically, our approach guarantees 100% constraint satisfaction in a network's output.

Introduction

An emerging theme in the development of deep learning is to provide expressive tools that allow domain experts to encode their prior knowledge into the training of neural networks. For example, in a manufacturing setting, we may wish to encode that an actuator for a robotic arm does not exceed some threshold (e.g., causing the arm to move at a hazardous speed). Another example is a self-driving car, where a controller should be known to operate within a predefined set of constraints (e.g., the car should always stop completely at a stop street). In such *safety critical* domains, machine learning solutions must guarantee to operate within distinct boundaries that are specified by experts (Amodei et al. 2016).

One possible solution is to encode the relevant domain knowledge directly into a network's architecture which may require non-trivial and/or domain-specific engineering (Goodfellow et al. 2016). An alternative approach is to express domain knowledge as logical constraints which can then be used to train neural networks (Xu et al. 2018; Fischer et al. 2019; Allen, Balažević, and Hospedales 2020). These approaches compile the constraints into the loss function of the training algorithm, by quantifying the extent to which the output of the network violates the constraints. This is appealing as logical constraints are easy to elicit from people. However, the solution outputted by the network is designed to minimize the loss function — which combines both data and constraints — rather than to guarantee the satisfaction of the domain constraints. Thus, representing constraints in the loss function is not suitable for safety critical domains where 100% constraint satisfaction is desirable.

Safety critical settings are not the only application for domain constraints. Another common problem in the training of large networks is that of data inefficiency. Deep models have shown unprecedented performance on a wide variety of tasks but these come at the cost of large data requirements.[1] For tasks where domain knowledge exists, learning algorithms should also use this knowledge to structure a network's training to reduce the data burden that is placed on the learning process (Fischer et al. 2019).

This paper directly addresses these challenges by providing a new way of representing domain constraints directly in the output layer of a network that guarantees constraint satisfaction. The proposed approach represents domain knowledge as a logical formula in disjunctive normal form (DNF). It augments the output layer of an existing neural network to include a separate transformation for each term in the DNF formula. We introduce a latent Categorical variable that selects the best transformation that optimizes the loss function of the data. In this way, we are able to represent arbitrarily complex domain constraints in an automated manner, and we are also able to guarantee that the output of the network satisfies the specified constraints.

We show the efficacy of this MultiplexNet approach in three distinct experiments. First, we present a density estimation task on synthetic data. It is a common goal in machine

[1] For instance OpenAI's GPT-3 (Brown et al. 2020) was trained on about 500 billion tokens and ImageNet-21k, used to train the ViT network (Dosovitskiy et al. 2020), consists of 14 million images.

learning to draw samples from a target distribution, and deep generative models have shown to be flexible and powerful tools for solving this problem. We show that by including domain knowledge, a model can learn to approximate an unknown distribution on fewer samples, and the model will (by construction) only produce samples that satisfy the domain constraints. This experiment speaks to both the data efficiency and the guaranteed constraint satisfaction desiderata. Second, we present an experiment on the popular MNIST data set (LeCun, Cortes, and Burges 2010) which combines structured data with domain knowledge. We structure the digits in a similar manner to the MNIST experiment from Manhaeve et al. (2018); however, we train the network in an entirely label-free manner (Stewart and Ermon 2017). In our third experiment, we apply our approach to the well known image classification task on the CIFAR100 data set (Krizhevsky, Hinton et al. 2009). Images are clustered according to "super classes" (e.g., both *maple tree* and *oak tree* fall under the super class *tree*). We follow the example of Fischer et al. (2019) and show that by including the knowledge that images within a super class are related, we can increase the classification accuracy at the super class level.

The paper contributes a novel and general way to integrate domain knowledge in the form of a logical specification into the training of neural networks. We show that domain knowledge may be used to restrict the network's operating domain such that any output is guaranteed to satisfy the constraints; and in certain cases, the domain knowledge can help to train the networks on fewer samples of data.

Problem Specification

We consider a data set of N i.i.d. samples from a hybrid (some mixture of discrete and/or continuous variables) probability density (Belle, Passerini, and Van den Broeck 2015). Moreover, we assume that: (1) the data set was generated by some random process $p^*(x)$; and (2) there exists domain or expert knowledge, in the form of a logical formula Φ, about the random process $p^*(x)$ that can express the domain where $p^*(x)$ is feasible (non-zero). Both of these assumptions are summarised in Eq. 1. In Eq. 1, the notation $x \models \Phi$, denotes that the sample x satisfies the formula Φ (Barrett et al. 2009). For example, if $\Phi := x > 3.5 \wedge y > 0$, and given some sample $(x, y) = (5, 2)$, we denote: $(x, y) \models \Phi$.

$$x \sim p^*(x) \implies x \models \Phi \tag{1}$$

Our aim is to approximate $p^*(x)$ with some parametric model $p_\theta(x)$ and to incorporate the domain knowledge Φ into the maximum likelihood estimation of θ, on the available data set.

Given knowledge of the constraints Φ, we are interested in ways of integrating these constraints into the training of a network that approximates $p^*(x)$. We desire an algorithm that does not require novel engineering to solve a reparameterisation of the network and moreover, especially salient for safety-critical domains, any sample x from the model, $x \sim p_\theta(x)$, should imply that the constraints are satisfied. This is an especially important aspect to consider when comparing this method to alternative approaches, namely Fischer et al. (2019) and Xu et al. (2018), that do not give this same guarantee.

Related Work

The integration of domain knowledge into the training of neural networks is an emerging area of focus. Many previous studies attempt to translate logical constraints into a numerical loss. The two most relevant works in this line are the DL2 framework by Fischer et al. (2019) and the Semantic Loss approach by Xu et al. (2018). DL2 uses a loss term that trades off data with the domain knowledge. It defines a non-negative loss by interpreting the logical constraints using fuzzy logic and defining a measure that quantifies how far a network's output is from the nearest satisfying solution. Semantic Loss also defines a term that is added to the standard network loss. Their loss function uses weighted model counting (Chavira and Darwiche 2008) to evaluate the probability that a sample from a network's output satisfies some Boolean constraint formulation. We differ from both of these approaches in that we do not add a loss term to the network's loss function, rather we compile the constraints directly into its output. Furthermore, in contrast to the works above, any network output from MultiplexNet will satisfy the domain constraints, which is crucial in safety critical domains.

It is also important to compare the expressiveness of the MultiplexNet constraints to those permitted by Fischer et al. (2019) and Xu et al. (2018). In MultiplexNet, the constraints can consist of any quantifier-free linear arithmetic formula over the rationals. Thus, variables can be combined over $+$ and $\geq$, and formulae over $\neg$, $\vee$ and $\wedge$. For example, $(x+y \geq 5) \wedge \neg(z \geq 5)$ but also $(x+y \geq z) \wedge (z > 5 \vee z < 3)$ are well defined formulae and therefore well defined constraints in our framework. The expressiveness is significant — for example, Xu et al. (2018) only allow for Boolean variables over $\{\neg, \wedge, \vee\}$. While Fischer et al. (2019) allow non-Boolean variables to be combined over $\{\geq, \leq\}$ and formulae to be used over $\{\neg, \vee, \wedge\}$, it is not a probabilistic framework, but one that is based on fuzzy logic. Thus, our work is probabilistic like the Semantic Loss (Xu et al. 2018), but it is more expressive in that it also allows real-valued variables over summations too.

Hu et al. (2016) introduce "iterative rule knowledge distillation" which uses a student and teacher framework to balance constraint satisfaction on first order logic formulae with predictive accuracy on a classification task. During training, the student is used to form a constrained teacher by projecting its weights onto a subspace that ensures satisfaction of the logic. The student is then trained to imitate the teacher's restricted predictions. Hu et al. (2016) use soft logic (Bach et al. 2017) to encode the logic, thereby allowing gradient estimation; however, the approach is unable to express rules that constrain real-valued outputs. Xsat (Fu and Su 2016) focuses on the Satisfiability Modulo Theory (SMT) problem, which is concerned with deciding whether a (usually a quantifier-free form) formula in first-order logic is satisfied against a background arithmetic theory; similar to what we consider. They present a means for solving SMT formulae but this is not differentiable. Manhaeve et al. (2018) present a compelling method for integrating logical constraints, in the

form of a ProbLog program, into the training of a network. However, the networks are embedded into the logic (represented by a Sentential Decision Diagram (Darwiche 2011)), as "neural predicates" and thus it is not clear how to handle the real-valued arithmetic constraints that we represent in MultiplexNet.

We also relate to work on program synthesis (Solar-Lezama 2009; Jha et al. 2010; Feng et al. 2017; Osera 2019) where the goal is to produce a valid program for a given set of constraints. Here, the output of a program is designed to meet a given specification. These works differ from this paper as they don't focus on the core problem of aiding training with the constraints and ensuring that the constraints are fully satisfied.

Other recent works have also explored how human expert knowledge can be used to guide a network's training. Ross and Doshi-Velez (2018); Ross, Hughes, and Doshi-Velez (2017) explore how the robustness of an image classifier can be improved by regularizing input gradients towards regions of the image that contain information (as identified by a human expert). They highlight the difficulty in eliciting expert knowledge from people but their technique is similar to the other works presented here in that the knowledge loss is still represented as an additive term to the standard network loss. Takeishi and Kawahara (2020) present an example of how the knowledge of relations of objects can be used to regularise a generative model. Again, the solution involves appending terms to the loss function, but they demonstrate that relational information can aid a learning algorithm. Alternative works have also explored means for constraining the latent variables in a latent variable model (Ganchev et al. 2010; Graça, Ganchev, and Taskar 2007). In contrast to this, we focus on constraining the output space of a generative model, rather than the latent space.

Finally, we mention work on the post-hoc verification of networks. Examples include the works of Katz et al. (2017) and Bunel et al. (2018) who present methods for validating whether a network will operate within the bounds of pre-defined restrictions. Our own work focuses on how to guarantee that the networks operate within given constraints, rather than how to validate their output.

Incorporating Constraints into Model Design

We begin by describing how a satisfiability problem can be hard coded into the output of a network. We then present how any specification of knowledge can be compiled into a form that permits this encoding. An overview of the proposed architecture with a general algorithm that details how to incorporate domain constraints into training a network can be found in Section: Architecture Overview of MultiplexNet in the supplementary material.

Satisfiability as Reparameterisation

Let $\tilde{x}$ denote the unconstrained output of a network. Let g be a network activation that is element-wise non-negative (for example an exponential function, or a ReLU (Nair and Hinton 2010) or Softplus (Dugas et al. 2001) layer). If the property to be encoded is a simple inequality $\Phi : \forall x\ cx \geq b$, it is sufficient to constrain $\tilde{x}$ to be non-negative by applying g and thereafter applying a linear transformation f such that: $\forall \tilde{x} : cf(g(\tilde{x})) \geq b$. In this case, f can implement the transformation $f(z) = sgn(c)z + \frac{b}{c}$ where sgn is the operator that returns the sign of c. By construction we have:

$$f(g(\tilde{x})) \models \Phi \tag{2}$$

It follows that more complex conjunctions of constraints can be encoded by composing transformations of the form presented in Eq. 2. We present below a few examples to demonstrate how this can be achieved for a number of common constraints (where $\tilde{x}$ always refers to the unconstrained output of the network):

$$a < x < b \ \rightarrow \ x = -g(-g(\tilde{x}) + k(a,b)) + b \tag{3}$$

$$x = c \ \rightarrow \ x = c \tag{4}$$

$$x_2 > h(x_1) \ \rightarrow \ x_1 = \tilde{x_1} \ ; \ x_2 = h(x_1) + g(\tilde{x_2}) \tag{5}$$

In Eq 3, we introduce the function $k(a, b)$. This is merely a function to compute the correct offset for a given activation g. In the case of the Softplus function, which is the function used in all of our experiments, $k(a, b) = log(exp(b-a)-1)$.

In Section: Experiments, we implement three varied experiments that demonstrate how complex constraints can be constructed from this basic primitive in Eq. 2. Conceptually, appending additional conjunctions to Φ serves to restrict the space that the output can represent. However, in many situations domain knowledge will consist of complicated formulae that exist well beyond mere conjunctions of inequalities.

While conjunctions serve to restrict the space permitted by the network's output, disjunctions serve to increase the permissible space. For two terms ϕ_1 and ϕ_2 in $\phi_1 \vee \phi_2$ there exist three possibilities: namely, that $x \models \phi_1$ or $x \models \phi_2$ or $(x \models \phi_1) \wedge (x \models \phi_2)$. Given the fact that any unconstrained network output can be transformed to satisfy some term ϕ_k, we propose to introduce multiple transformations of a network's unconstrained output, each to model the different terms ϕ_k. In this sense, the network's output layer can be viewed as a multiplexor in a logical circuit that permits for a branching of logic. If $h_1(\tilde{x})$ represents the transformation of $\tilde{x}$ that satisfies ϕ_1 and $h_2(\tilde{x}) \models \phi_2$ then we know the output must also satisfy $\phi_1 \vee \phi_2$ by choosing either h_1 or h_2. It is this branching technique for dealing with disjunctions that gives rise to the name of the approach: MultiplexNet.

We finally turn to the desideratum of allowing any Boolean formula over linear inequalities as the input for the domain constraints. The suggested approach can represent conjunctions of constraints and disjunctions between these conjunctive terms, which is exactly a DNF representation. Thus, the approach can be used with any transformed version of Φ that is in DNF (Darwiche and Marquis 2002). We propose to use an off-the-shelf solver, e.g., Z3 (De Moura and Bjørner 2008), to provide the logical input to the algorithm that is in DNF. We thus assume the domain knowledge Φ is expressed as:

$$\Phi = \phi_1 \vee \phi_2 \vee \ldots \vee \phi_k \tag{6}$$

If h_k is the branch of MultiplexNet that ensures the output of the network $x \models \phi_k$ then it follows by construction that

$h_k(\tilde{x}) \models \Phi$ for all $k \in [1, \ldots, K]$. For example, consider a network with a single real-valued output $\tilde{x} \in \mathbb{R}$. If the knowledge $\Phi := (x \geq 2) \vee (x \leq -2)$, we would then have the two terms $h_1(\tilde{x}) = g(\tilde{x}) + 2$ and $h_2(\tilde{x}) = -g(-\tilde{x}) - 2$. Here, g is the network activation that is element-wise non-negative that was referred to in Section: Satisfiability as Reparameterisation. It is clear that both $x_1 = h_1(\tilde{x})$ and $x_2 = h_2(\tilde{x})$ satisfy the formula Φ.

Lemma 0.1 *Suppose Φ is a quantifier free first-order formula in DNF over $\{x_1, \ldots, x_J\}$ consisting of terms $\phi_1 \vee \ldots \vee \phi_K$. Since each branch of MultiplexNet (h_k) is constructed to satisfy a specific term (ϕ_k), by construction, the output of MultiplexNet will satisfy Φ: $\{\hat{x}_1, \ldots, \hat{x}_J\} \models \Phi$.*

MultiplexNet as a Latent Variable Problem

MultiplexNet introduces a latent Categorical variable k that selects among the different terms $\phi_k, k \in [1, \ldots, K]$. The model then incorporates a constraint transformation term h_k conditional on the value of the Categorical variable.

$$p_\theta(x) = p_\theta(h_k(x)|k)p(k) \tag{7}$$

A lower bound on the likelihood of the data can be obtained by introducing a variational approximation to the latent Categorical variable k. This standard form of the variational lower bound (ELBO) is presented in Eq. 8.

$$\begin{aligned} \log p_\theta(x) &\geq \mathbb{E}_{q(k)}[\log p_\theta(h_k(x)|k) + \log p(k) - \log q(k)] \\ &:= ELBO(x) \end{aligned} \tag{8}$$

Gradient based methods require calculating the derivative of Eq. 8. However, as $q(k)$ is a Categorical distribution, the standard reparameterisation trick cannot be applied (Kingma and Welling 2014). One possibility for dealing with this expectation is to use the score function estimator, as in REINFORCE (Williams 1992); however, while the resulting estimator is unbiased, it has a high variance (Mnih and Gregor 2014). It is also possible to replace the Categorical variable with a continuous approximation as is done by Maddison, Mnih, and Teh (2017) and Jang, Gu, and Poole (2016); or, if the dimensionality of the Categorical variable is small, it can be marginalised out as in (Kingma et al. 2014). In the experiments in Section: Experiments, we follow Kingma et al. (2014) and marginalise this variable.[2] leading to the following learning objective:

$$\begin{aligned} \mathcal{L}(\theta; x) = -\sum_{k=1}^{K} q(k) \big[&\log p_\theta(h_k(x)|k) \\ &+ \log p(k) - \log q(k) \big] \end{aligned} \tag{9}$$

We show in Section: Experiments that this approach can be applied equally successfully for a generative modeling task (where the goal is density estimation) as for a discriminative task (where the goal is structured classification). This helps to demonstrate the universal applicability of incorporating domain knowledge into the training of networks.

[2] Although we note that the alternatives should also be explored.

Architecture Overview of MultiplexNet

MultiplexNet accepts as input a data set consisting of samples from some target distribution, $p^*(x)$, and some constraints, Φ that are known about the data set. We assume that the constraints are correct, in that Eq. 1 holds for all x. We aim to model the unknown density, p^*, by maximising the likelihood of a parameterised model, $p_\theta(x)$ on the given data set. Moreover, our goal is to incorporate the domain constraints, Φ, into the training of this model.

For each term ϕ_k in the DNF representation of $\Phi = \phi_1 \vee \phi_2 \vee \cdots \vee \phi_K$, we introduce a transformation, h_k, that ensures any real-valued input is transformed to satisfy that term. With an activation, g, that is element-wise non-negative, we can suitably restrict the domain of any real-valued variable such that the output satisfies ϕ_k. For example, consider the constraints, e.g., $\phi_1 = (x > y + 2) \wedge (x < 5)$ and assuming a Softplus activation. The transformation $h_1(x') = -g(-(g(x') + \alpha) + \beta)$ will constrain the real-valued variable x' such that ϕ_1 is satisfied. In this example, y does not need to be constrained. Here $\beta = 5$ and $\alpha = \log(e^{5-(y+2)} - 1)$. Any combination of inequalities can be suitably restricted in this way. Equality constraints can be handled by setting the output to the value that is specified.

MultiplexNet therefore accepts the unconstrained output of a network, $x' \in \mathbb{R}$, and introduces K constraint terms h_k that each guarantee the constrained output $x_k = h(x')$ will satisfy a term, ϕ_k, in the DNF representation of the constraints, Φ. The output of the network is then K transformed versions of x' where each output x_k is guaranteed to satisfy Φ. The Categorical selection variable, k, can be marginalised out leading to the objective presented in Eq. 9.

If we refer to the original log-likelihood of the unconstrained model as $\mathcal{L}'$ and the entropy of the categorical approximation as $\mathcal{H}(q(k \mid x))$ then Eq. 9 can be simplified as in Eq. 10. Note that we assume here a constant prior on the Categorical variable k and x_k is the k^{th} constrained term of the unconstrained output of the network: $x_k = h_k(x')$.

$$\mathcal{L}(\theta) = \sum_{i=1}^{K} q(k \mid x)\mathcal{L}'(x_k) + \mathcal{H}(q(k \mid x)) \tag{10}$$

This architecture is represented pictorially in Fig. 1.

Experiments

We apply MultiplexNet to three separate experimental domains. The first domain demonstrates a density estimation task on synthetic data when the number of available data samples are limited. We show how the value of the domain constraints improves the training when the number of data samples decreases; this demonstrates the power of adding domain knowledge into the training pipeline. The second domain applies MultiplexNet to labeling MNIST images in an unsupervised manner by exploiting a structured problem and data set. We use a similar experimental setup to the MNIST experiment from DeepProbLog (Manhaeve et al. 2018); however, we present a natural integration with a generative model that is not possible with DeepProbLog. The third experiment uses hierarchical domain knowledge to facilitate an image

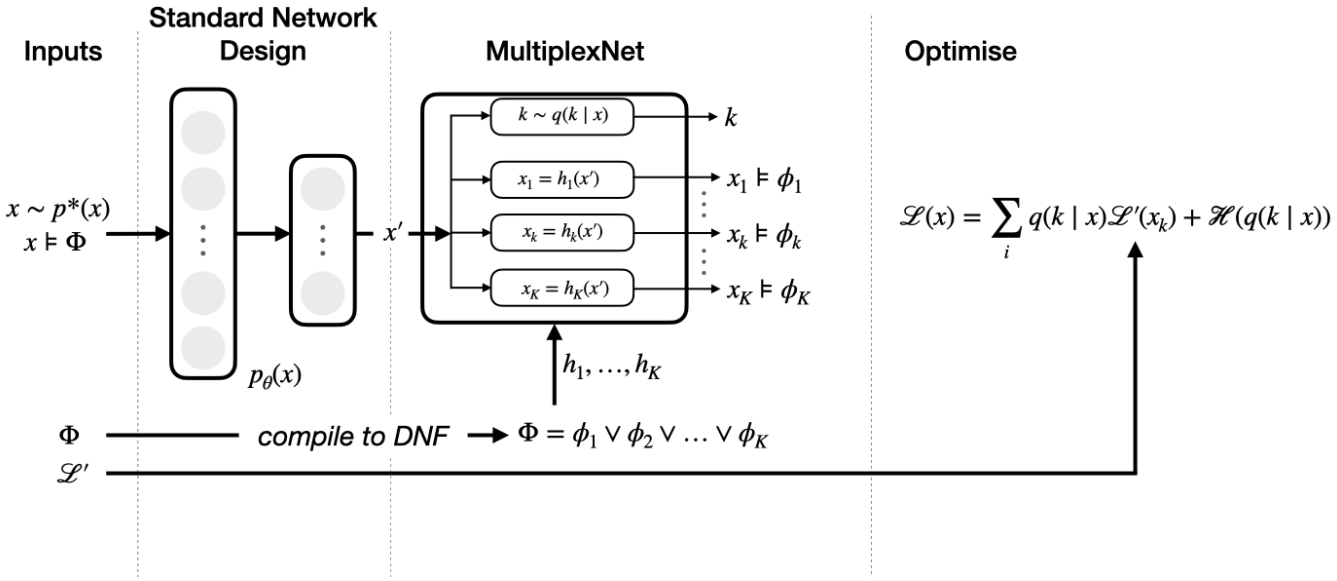


Figure 1: Architecture of the MultiplexNet. We show how to append this framework to an existing learning scheme. The unconstrained output of the network x', along with the constrain transformation terms $h_1, \ldots, h_K$ are used to create K constrained output terms $x_1, \ldots, x_K$. The latent Categorical variable k is used to select which term is active for a given input. In this paper, we marginalise the Categorical variable leading to the specified loss function.

classification task taken from Fischer et al. (2019) and Xu et al. (2018). We show how the use of this knowledge can help to improve classification accuracy at the super class level.

Synthetic Data

In this illustrative experiment, we consider a target data set that consists of the six rectangular modes that are shown in Figure 2. The samples from the true target density are shown, along with 8 rectangular boxes in red. The rectangular boxes represent the domain constraints for this experiment. Here, we show that an expert might know where the data can exist but that the domain knowledge does not capture all of the details of the target density. Thus, the network is still tasked with learning the intricacies of the data that the domain constraints fail to address (e.g., not all of the area within the constraints contains data). However, we desire that the knowledge leads the network towards a better solution, and also to achieve this on fewer data samples from the true distribution.

This experiment represents a density estimation task and thus we use a likelihood-based generative model to represent the unknown target density, using both data samples and domain knowledge. We use a variational autoencoder (VAE) which optimizes a lower bound to the marginal log-likelihood of the data. However, a different generative model, for example a normalizing flow (Papamakarios et al. 2019) or a GAN (Goodfellow et al. 2014), could as easily be used in this framework. We optimize Eq. 9 where, for this experiment, the likelihood term $\log p_\theta(\cdot \mid k)$ is replaced by the standard VAE loss. Additional experimental details, as well as the full

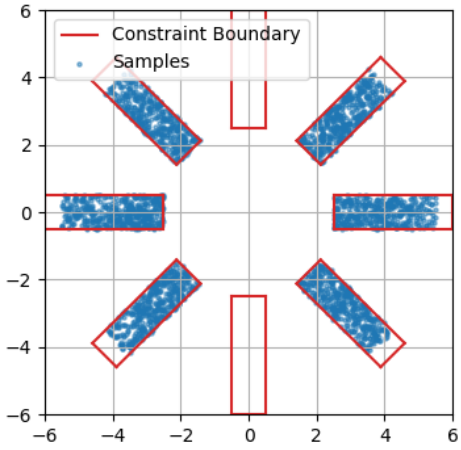
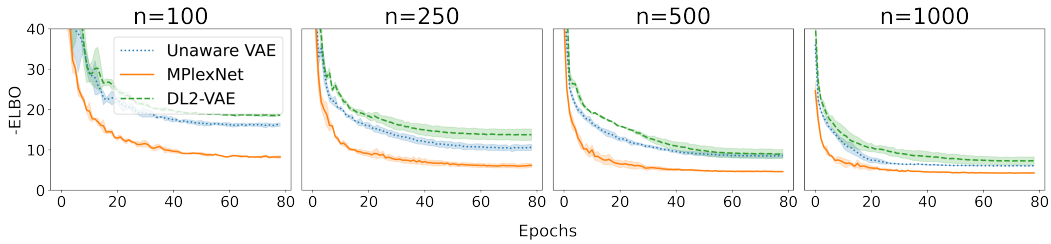
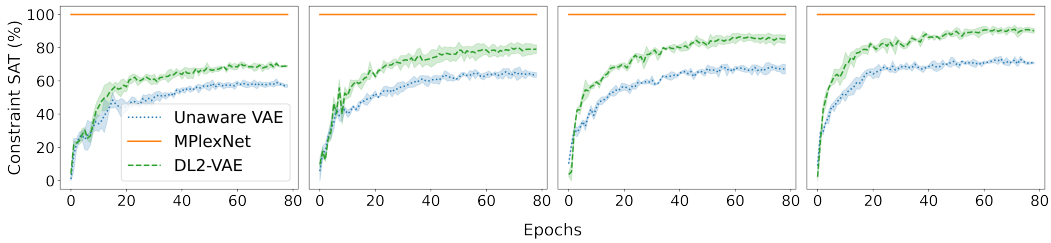


Figure 2: Simulated data from an unknown density. We assume that we know some constraints about the domain; these are represented by the red boxes. We aim to represent the unknown density, subject to the knowledge that the constraints must be satisfied.



(a)



(b)

Figure 3: Results from the synthetic data experiment (a) Negative lower bound to the held out likelihood of data (-ELBO). The MultiplexNet approach learns to represent the data with a higher likelihood, and faster than the baselines. (b) % of reconstruction samples from the VAE that obey the domain constraints. The MultiplexNet approach, by construction, can only generate samples within the specified constraints.

loss function, can be found in Appendix A.

We vary the size of the training data set with $N \in \{100, 250, 500, 1000\}$ as the four experimental conditions. We compare the lower bound to the marginal log-likelihood under three conditions: the MultiplexNet approach, as well as two baselines. The first baseline (Unaware VAE) is a vanilla VAE that is unaware of the domain constraints. This scenario represents the standard setting where domain knowledge is simply ignored in the training of a deep generative network. The second baseline (DL2-VAE) represents a method that appends a loss term to the standard VAE loss. It is important to note that this approach, from DL2 (Fischer et al. 2019), does not guarantee that the constraints are satisfied (clearly seen in Figure 3b).

Figure 3 presents the results where we run the experiment on the specified range of training data set sizes. The top plot shows the variational loss as a function of the number of epochs. For all sizes of training data, the MultiplexNet loss on a test set can be seen to outperform the baselines. By including domain knowledge, we can reach a better result, and on fewer samples of data, than by not including the constraints. More important than the likelihood on held-out data is that the samples from the models' posterior should conform with the constraints. Figure 3b shows that the baselines struggle to learn the structure of the constraints. While the MultiplexNet solution is unsurprising, the constraints are followed by construction, the comparison to the baselines is stark. We also present samples from both the prior and the posterior for all of these models in Appendix A. In all of these, MultiplexNet learns to approximate the unknown density within the predefined boundaries of the provided constraints.

MNIST - Label-free Structured Learning

We demonstrate how a structured data set, in combination with the relevant domain knowledge, can be used to make novel inferences in that domain. Here, we use a similar experiment to that from Kingma et al. (2014) where we model the MNIST digit data set in an unsupervised manner. Moreover, we take inspiration from Manhaeve et al. (2018) for constructing a structured data set were the images represent the terms in a summation (e.g., $image(2) + image(3) = 5$). However, we add to the complexity of the task by (1) using no labels for any of the images;[3] and, (2) considering a generative task.

Kingma et al. (2014) propose a generative model that reasons about the cluster assignment of a data point (a single image). In particular, in their popular "Model 2," they describe a generative model for an image x such that the probability of the image pixel values are conditioned on a latent variable (z) and a class label (y): $p_\theta(x \mid z, y)p(z \mid y)p(y)$. We can interpret this model using the MultiplexNet framework where the cluster assignment label $y = k$ implies that the image x was generated from cluster k. Given a reconstruction loss for image x, conditioned on class label y ($\mathcal{L}(x, y)$), the domain knowledge in this setting is: $\Phi := \bigvee_{k=1}^{10} \mathcal{L}(x, y) \wedge (y = k)$. We can successfully model the clustering of the data using this setup but there is no means for determining which label

[3] In the MNIST experiment from Manhaeve et al. (2018), the authors use the result of the summation as labels for the algorithm. We have no such analogy in this experiment and thus cannot use their DeepProbLog implementation as a baseline.

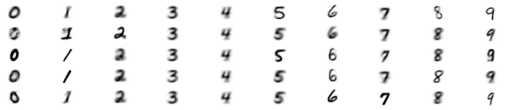


Figure 4: Reconstructed/Decoded samples from the prior, z, of the trained model where each column conditions on a different value for y. It can be seen that the model has learnt to represent all of the digits $[0-9]$ with the correct class label, even though no labels were supplied to the training process.

corresponds to which cluster assignment.

We therefore propose to augment the data set such that each input is a quintuple of four images (x_1, x_2, x_3, x_4) in the form $label(x_1) + label(x_2) = (label(x_3), label(x_4))$. Here, the inputs $label(x_1)$ and $label(x_2)$ can be any integer from 0 to 9 and the result $(label(x_3), label(x_4))$ is a two digit number from (00) to (18). While we do not know explicitly any of the cluster labels, we do know that the data conform to this standard. Thus for all i, j, k where $k = i + j$, the domain knowledge is of the form:

$$\Phi := \bigvee_{i,j,k} \Big[(y_1 = i) \wedge (y_2 = j) \wedge (y_3 = \mathbb{1}_{k>9}) \\ \wedge (y_4 = k \bmod 10) \bigwedge_{n=1}^{4} \mathcal{L}(x_n, y_n) \Big] \tag{11}$$

In this setting, the categorical variable in the MultiplexNet chooses among the 100 combinations that satisfy $label(x_1) + label(x_2) = (label(x_3), label(x_4))$. This experiment has similarities to DeepProbLog (Manhaeve et al. 2018) as the primitive $\mathcal{L}(x, y)$ is repeated for each digit. In this sense, it is similar to the "neural predicate" used by Manhaeve et al. (2018), and the MultiplexNet output layer implements what would be the logical program from DeepProbLog. However, it is not clear how to implement this label-free, generative task within the DeepProbLog framework.

In Figure 4, we present samples from the prior, conditioned on the different class labels. The model is able to learn a class-conditional representation for the data, *given no labels for the images*. This is in contrast to a vanilla model (from Kingma et al. (2014)) which does not use the structure of the data set to make inferences about the class labels. We present these baseline samples as well as the experimental details and additional notes in Appendix A. Empirically, the results from this experiment were sensitive to the network's initialisation and thus we report the accuracy of the top 5 runs. We selected the runs based on the loss (the ELBO) on a validation set (i.e., the labels were still not used in selecting the run). The accuracy of the inferred labels on held out data is 97.5 ± 0.3.

Hierarchical Domain Knowledge on CIFAR100

The final experiment demonstrates how to encode hierarchical domain knowledge into the output layer of a network. The CIFAR100 (Krizhevsky, Hinton et al. 2009) data set consists of 100 classes of images where the 100 classes are in turn broken into 20 super-classes (SC). We wish to encode the belief that images from the same SC are semantically related. Following the encoding in Fischer et al. (2019), we consider constraints which specify that groups of classes should together be very likely or very unlikely. For example, suppose that the SC label is *trees* and the class label is *maple*. Our domain knowledge should state that the *trees* group must be very likely even if there is uncertainty in the specific label *maple*. Intuitively, it is egregious to misclassify this example as a *tractor* but it would be acceptable to make the mistake of *oak*. This can be implemented by training a network to predict first the SC for an unknown image and thereafter the class label, conditioned on the value for the SC.

We chose rather to implement this same knowledge using the MultiplexNet framework. Let $x_k \in SC_i$ denote the output of a network that predicts the k^{th} class label within the i^{th} SC. Let $\alpha \in [0, 1]$ denote the minimum requirement for a SC prediction (e.g., if $\alpha = 0.95$, we require that a SC be predicted with probability 0.95 or more). The domain knowledge is:

$$\bigvee_{i=1}^{20} \left[\bigwedge_{k \in SC_i} \left(x_k > \log(\frac{\alpha}{1-\alpha}) + \log \sum_{j \notin SC_i} \exp\{x_j\} \right) \right] \tag{12}$$

Eq. 12 states that for all labels within a SC group, the unnormalised logits of the network should be greater than the normalised sum of the other labels belonging to the other SCs with a margin of $\log(\frac{\alpha}{1-\alpha})$. We explain Eq. 12 further and present other experimental details in Appendix A. This constraint places a semantic grouping on the data as the network is forced into a low entropy prediction at the super class level.

We compare the performance of MultiplexNet to three baselines and report the prediction accuracy on the fine class label as well that on the super class label. We use a Wide ResNet 28-10 (Zagoruyko and Komodakis 2016) model in all of the experimental conditions. The first two baselines (Vanilla) only use the Wide ResNet model and are trained to predict the fine class and the super class labels respectively. The second baseline (Hierarchical) is trained to predict the super class label and thereafter the fine class label, conditioned on the value for the super class label. This represents the bespoke engineering solution to this hierarchical problem. The final baseline (DL2) implements the same logical specification that is used for MultiplexNet but uses the DL2 framework to append to the standard cross-entropy loss function.

Table 1 presents the results for this experiment. Firstly, it is important to note the difficulty of this task. The Vanilla ResNet that predicts only the super-class labels for the images under performs the baseline that is tasked with predicting the true class label. Moreover, while the hierarchical baseline does outperform the vanilla models on the task of super-class prediction, this comes at a cost to the true class accuracy. As the hierarchical baseline represents the bespoke engineering

Table 1: Accuracy on class label prediction and super-class label prediction, and constraint satisfaction on CIFAR100 data set

Model	Class Accuracy	Super-class Accuracy	Constraint Satisfaction
Vanilla ResNet	$75.0 \pm (0.1)$	$84.0 \pm (0.2)$	$83.8 \pm (0.1)$
Vanilla ResNet (SC only)	NA	$83.2 \pm (0.2)$	NA
Hierarchical Model	$71.2 \pm (0.2)$	$84.7 \pm (0.1)$	$\mathbf{100.0 \pm (0.0)}$
DL2	$\mathbf{75.3 \pm (0.1)}$	$84.3 \pm (0.1)$	$85.8 \pm (0.2)$
MultiplexNet	$74.4 \pm (0.2)$	$\mathbf{85.4 \pm (0.3)}$	$\mathbf{100.0 \pm (0.0)}$

solution to the problem, it also achieves 100% constraint satisfaction, but this comes at the cost of domain specific and custom implementation. The MultiplexNet approach provides a slight improvement at the SC classification accuracy and importantly, the domain constraints are always met. As the domain knowledge prioritizes the accuracy at the SC level, we note that the MultiplexNet approach does not outperform the Vanilla ResNet at the class accuracy. Surprisingly, the DL2 baseline improves upon the class accuracy but it has a limited impact on the super class accuracy and on the constraint satisfaction.

Limitations and Discussion

The limitations of the suggested approach relate to the technical specification of the domain knowledge and to the practical implementation of this knowledge. We discuss first these two aspects and then we discuss a potential negative societal impact.

First, we require that experts be able to express precisely, in the form of a logical formula, the constraints that are valid for their domain. This may not always be possible. For example, given an image classification task, we may wish to describe our knowledge about the *content* of the images. Consider an example where images contain pictures of *dogs* and *fish* and that we wish to express the knowledge that dogs have four legs and fish have gills. It is not clear how these conceptual constraints would then be mapped to a pixel level for actual specification. Moreover, it is entirely plausible to have images of dogs that do not include their legs, or images of fish where, for example, we only see their tails. The logical statement itself is brittle in these instances and would serve to hinder the training, rather than to help it. This example serves to present the inherent difficulty that is present when actually expressing robust domain knowledge in the form of logical formulae.

The second major limitation of this approach deals with the DNF requirement on the input formula. We require that knowledge be expressed in this form such that the "or" condition is controlled by the latent Categorical variable of MultiplexNet. It is well known that certain formulae have worst case representations in DNF that are exponential in the number of variables. This could be undesirable in that the network would have to learn to choose among the exponentially many terms. Although there is ample research in Knowledge Representation and SAT communities on bench-marking theories and constraints where conversions to certain normal forms leads to such exponential blow ups in length, the literature on regularization-based logic approaches for neural networks is far less mature. As the field matures, we expect alternative normal forms could be used for different types of problems, however, the choice of normal form will always limit the application of the approach to certain problems.

One of the overarching motivations for this work is to constrain networks for safety critical domains. While constrained operation might be desired on many accounts, there may exist edge cases where an autonomously acting agent should act in an undesirable manner to avoid an even more undesirable outcome (a thought experiment of this spirit is the well known Trolley Problem (Hammond and Belle 2021)). By guaranteeing that the operating conditions of a system be restricted to some range, our approach does encounter vulnerability with respect to edge, and unforeseen, cases. However, to counter this point, we argue it is still necessary for experts to define the boundaries over the operation domain of a system in order to explicitly test and design for known worst case scenario settings.

Conclusions and Future Work

This work studied how logical knowledge in an expressive language could be used to constrain the output of a network. It provides a new and general way to encode domain knowledge as logical constraints directly in the output layer of a network. Compared to alternative approaches, we go beyond propositional logic by allowing for arithmetic operators in our constraints. We are able to guarantee that the network output is 100% compliant with the domain constraints, which the alternative approaches, which append a "constraint loss," are unable to match. Thus our approach is especially relevant for safety critical settings in which the network must guarantee to operate within predefined constraints. In a series of experiments we demonstrated that our approach leads to better results in terms of data efficiency (the amount of training data that is required for good performance), reducing the data burden that is placed on the training process. In the future, we are excited about exploring the prospects for using this framework on downstream tasks, such as robustness to adversarial attacks.

Acknowledgements

Hoernle is funded by a Commonwealth Scholarship. Belle was supported by a Royal Society University Research Fellowship. Belle was also supported by a grant from the UKRI Strategic Priorities Fund to the UKRI Research Node on Trustworthy Autonomous Systems Governance and Regulation (EP/V026607/1, 2020-2024)

References

Allen, C.; Balažević, I.; and Hospedales, T. 2020. A Probabilistic Framework for Discriminative and Neuro-Symbolic Semi-Supervised Learning. *arXiv preprint arXiv:2006.05896*.

Amodei, D.; Olah, C.; Steinhardt, J.; Christiano, P.; Schulman, J.; and Mané, D. 2016. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*.

Bach, S. H.; Broecheler, M.; Huang, B.; and Getoor, L. 2017. Hinge-Loss Markov Random Fields and Probabilistic Soft Logic. *J. Mach. Learn. Res.*, 18(1): 3846–3912.

Barrett, C.; Sebastiani, R.; Seshia, S. A.; Tinelli, C.; Biere, A.; Heule, M.; van Maaren, H.; and Walsh, T. 2009. Handbook of satisfiability. *Satisfiability modulo theories*, 185: 825–885.

Belle, V.; Passerini, A.; and Van den Broeck, G. 2015. Probabilistic inference in hybrid domains by weighted model integration. In *Proceedings of 24th International Joint Conference on Artificial Intelligence (IJCAI)*, volume 2015, 2770–2776.

Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Bunel, R.; Turkaslan, I.; Torr, P. H. S.; Kohli, P.; and Mudigonda, P. K. 2018. A Unified View of Piecewise Linear Neural Network Verification. *Advances in Neural Information Processing Systems*, 4795–4804.

Chavira, M.; and Darwiche, A. 2008. On probabilistic inference by weighted model counting. *Artificial Intelligence*, 172(6-7): 772–799.

Darwiche, A. 2011. SDD: A new canonical representation of propositional knowledge bases. In *Twenty-Second International Joint Conference on Artificial Intelligence*.

Darwiche, A.; and Marquis, P. 2002. A knowledge compilation map. *Journal of Artificial Intelligence Research*, 17: 229–264.

De Moura, L.; and Bjørner, N. 2008. Z3: An efficient SMT solver. In *International conference on Tools and Algorithms for the Construction and Analysis of Systems*, 337–340. Springer.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Dugas, C.; Bengio, Y.; Bélisle, F.; Nadeau, C.; and Garcia, R. 2001. Incorporating second-order functional knowledge for better option pricing. *Advances in neural information processing systems*, 472–478.

Feng, Y.; Martins, R.; Wang, Y.; Dillig, I.; and Reps, T. W. 2017. Component-Based Synthesis for Complex APIs. In *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages*, 599–612.

Fischer, M.; Balunovic, M.; Drachsler-Cohen, D.; Gehr, T.; Zhang, C.; and Vechev, M. 2019. DL2: Training and Querying Neural Networks with Logic. In *Proceedings of the 36th International Conference on Machine Learning*, 1931–1941.

Fu, Z.; and Su, Z. 2016. XSat: A Fast Floating-Point Satisfiability Solver. In *Proceedings of the 28th International Conference on Computer Aided Verification, Part II*, 187–209. Springer. ISBN 978-3-319-41539-0.

Ganchev, K.; Graça, J.; Gillenwater, J.; and Taskar, B. 2010. Posterior regularization for structured latent variable models. *The Journal of Machine Learning Research*, 11: 2001–2049.

Goodfellow, I.; Bengio, Y.; Courville, A.; and Bengio, Y. 2016. *Deep learning*, volume 1. MIT press Cambridge.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, 2672–2680.

Graça, J. V.; Ganchev, K.; and Taskar, B. 2007. Expectation maximization and posterior constraints.

Hammond, L.; and Belle, V. 2021. Learning tractable probabilistic models for moral responsibility and blame. *Data Mining and Knowledge Discovery*, 35(2): 621–659.

Hu, Z.; Ma, X.; Liu, Z.; Hovy, E.; and Xing, E. 2016. Harnessing Deep Neural Networks with Logic Rules. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2410–2420.

Jang, E.; Gu, S.; and Poole, B. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Jha, S.; Gulwani, S.; Seshia, S. A.; and Tiwari, A. 2010. Oracle-Guided Component-Based Program Synthesis. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1*, 215–224.

Katz, G.; Barrett, C.; Dill, D. L.; Julian, K.; and Kochenderfer, M. J. 2017. Reluplex: An efficient SMT solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, 97–117. Springer.

Kingma, D. P.; Mohamed, S.; Rezende, D. J.; and Welling, M. 2014. Semi-supervised learning with deep generative models. In *Advances in neural information processing systems*, 3581–3589.

Kingma, D. P.; and Welling, M. 2014. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations (ICLR)*.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images.

LeCun, Y.; Cortes, C.; and Burges, C. 2010. MNIST handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2.

Maddison, C. J.; Mnih, A.; and Teh, Y. W. 2017. The concrete distribution: A continuous relaxation of discrete random variables. In *5th International Conference on Learning Representations (ICLR)*.

Manhaeve, R.; Dumancic, S.; Kimmig, A.; Demeester, T.; and De Raedt, L. 2018. Deepproblog: Neural probabilistic logic programming. *Advances in Neural Information Processing Systems*, 31: 3749–3759.

Mnih, A.; and Gregor, K. 2014. Neural variational inference and learning in belief networks. In *International Conference on Machine Learning*, 1791–1799. PMLR.

Nair, V.; and Hinton, G. E. 2010. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, 807–814.

Osera, P.-M. 2019. Constraint-Based Type-Directed Program Synthesis. In *Proceedings of the 4th ACM SIGPLAN International Workshop on Type-Driven Development*, 64–76.

Papamakarios, G.; Nalisnick, E.; Rezende, D. J.; Mohamed, S.; and Lakshminarayanan, B. 2019. Normalizing Flows for Probabilistic Modeling and Inference. *arXiv preprint arXiv:1912.02762*.

Ross, A.; and Doshi-Velez, F. 2018. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Ross, A. S.; Hughes, M. C.; and Doshi-Velez, F. 2017. Right for the Right Reasons: Training Differentiable Models by Constraining their Explanations. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI*, 2662–2670.

Solar-Lezama, A. 2009. The Sketching Approach to Program Synthesis. In *Proceedings of the 7th Asian Symposium on Programming Languages and Systems*, 4–13.

Stewart, R.; and Ermon, S. 2017. Label-free supervision of neural networks with physics and domain knowledge. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

Takeishi, N.; and Kawahara, Y. 2020. Knowledge-Based Regularization in Generative Modeling. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI*, 2390–2396.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4): 229–256.

Xu, J.; Zhang, Z.; Friedman, T.; Liang, Y.; and Van den Broeck, G. 2018. A Semantic Loss Function for Deep Learning with Symbolic Knowledge. In *Proceedings of the 35th International Conference on Machine Learning*, 5502–5511.

Zagoruyko, S.; and Komodakis, N. 2016. Wide Residual Networks. In *British Machine Vision Conference*.