Contextual Classification Using Self-Supervised Auxiliary Models for Deep Neural Networks

Sebastian Palacio[1,2], Philipp Engler[1,2], Jörn Hees[2], Andreas Dengel[1,2]
[1]DFKI GmbH
[2]TU Kaiserslautern
Kaiserslautern, Germany
firstname.lastname@dfki.de

***Abstract*—Classification problems solved with deep neural networks (DNNs) typically rely on a closed world paradigm, and optimize over a single objective (*e.g.*, minimization of the cross-entropy loss). This setup dismisses all kinds of supporting signals that can be used to reinforce the existence or absence of particular patterns. The increasing need for models that are interpretable by design makes the inclusion of said contextual signals a crucial necessity. To this end, we introduce the notion of Self-Supervised Autogenous Learning (SSAL). A SSAL objective is realized through one or more additional targets that are derived from the original supervised classification task, following architectural principles found in multi-task learning. SSAL branches impose low-level priors into the optimization process (*e.g.*, grouping). The ability of using SSAL branches during inference, allow models to converge faster, focusing on a richer set of class-relevant features. We equip state-of-the-art DNNs with SSAL objectives and report consistent improvements for all of them on CIFAR100 and Imagenet. We show that SSAL models outperform similar state-of-the-art methods focused on contextual loss functions, auxiliary branches and hierarchical priors.**

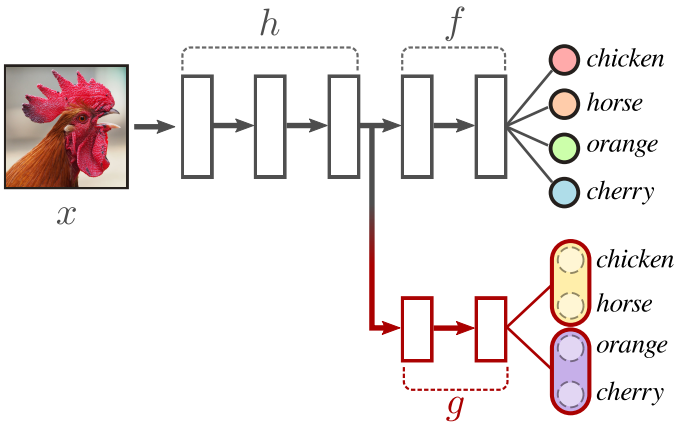

Fig. 1. Overview of SSAL models. Starting from a common feature representation h, a supervised goal f and an auxiliary branch g are used for training and prediction. Training objective for g is derived from the original labels used for f following a mutually exclusive grouping.

I. Introduction

Typical machine learning classification models are isolated in nature *i.e.*, they are defined, and operate under a closed world paradigm [1] where all possible inputs belong to one out of multiple but finite pre-defined classes. This simplification goes against emerging needs for more interpretable models [2], [3] potentially harming performance, as humans naturally rely on external, complementary knowledge to find corroborating or conflicting evidence for a particular decision. Our brains process information in a non-linear fashion, aggregating heterogeneous stimuli that converge to a unified interpretation or action. Closed world models are thereby semantically disconnected from the patterns we may deem reasonable, making the quest for explanations an ill-posed endeavor. The field of adversarial perturbations is a good example of this semantic gap for explanability [4]. Despite having input samples that preserve all the perceptually relevant information, adversarially perturbed samples can be misclassified with high probability. The effectiveness of adversarial attacks provides a strong body of evidence that patterns extracted by neural networks are effective but fundamentally different from the ones we are able to understand.

Is there a way to embed context signals into the training process of a neural network without resorting to additional ground-truth? Contextual information can be certainly collected alongside class labels, but what exactly should that context be, is non-trivial to define, and costly to gather. In this work, we propose the use of auxiliary classifiers to solve a surrogate objective that is still closely related to the original task. Intuitively, we design the auxiliary task based on a simple characteristic of independent classification problems: if a model can classify a set of disjoint fine-grained classes, it should also be able to classify an arbitrary grouping of those classes. The architecture of a traditional model can be hence modified by adding a symbiotic auxiliary classifier that shares a common feature representation, but optimizes the grouping objective instead (Figure 1). For prediction, a combination of both outputs is possible through an element-wise (Hadamard) product or via a learned linear combination.

We describe the auxiliary task as "*autogeonous*" as it is self-supervised *i.e.*, it does not rely on additional annotations, and it is derived from a source within the dataset, namely the original labels. We refer to the use of auxiliary classifiers using such a surrogate objective as *Self-Supervised Autogenous Learning* (SSAL). The close relationship between a SSAL task and the main task allows the main model to benefit from auxiliary classifiers both during training and inference.

The benefits of SSAL are two-fold: it acts as regularizer for the original architecture, and the contextual nature of the

surrogate objective provides supporting evidence that aligns with human expectations. Furthermore, we show through a set of comprehensive experiments on CIFAR100 [5], TinyImagenet [6], and Imagenet [7], that SSAL classifiers consistently yield superior accuracy even after controlling for model size.

II. Related Work

Early Work: The notion of auxiliary classifiers for neural networks can be tracked down back to the early days of machine learning. In 1990, Abu *et al.* [8] proposed the use of "hints" *i.e.*, additional knowledge about an objective for neural networks. These hints were represented as an additional gradient term for a Multi-Layer Perceptron trained via back-propagation [9]. They concluded that the use of hints could allow networks to converge faster as the set of potential solutions was constrained further.

Multi-Task Learning: This idea was later expanded to what is currently known as Multi-Task Learning (MTL) [10]. In this scenario, a network is trained jointly on multiple tasks that are not necessarily aligned but still share some commonalities, and therefore can benefit from a joint representation. However, we are not aware of any metric that quantifies how related two tasks are. Furthermore, the result of MTL is a model that can solve two problems but the heterogeneity of the domains often precludes the possibility of a joint prediction. Although recent advances on MTL have demonstrated that jointly learning disparate tasks does not harm performance [11], measuring the net benefit for all tasks still depends on a perceived ad-hoc similarity between them. The symbiotic interaction between the two objectives is thereby obscured, making it difficult to establish if one or both tasks benefit from one another (*i.e.*, if the relation is mutualistic, commensalistic or parasitic). Some surprising and rather unexpected relationships between tasks have been reported in the literature, showing that the degree of relatedness between tasks is not trivial to assess. For example Lee *et al.* [12] showed how a video classifier trained on finding the right order of a clip of shuffled frames was advantageous for fine-tuning on action recognition, image classification and object detection tasks. Similarly, Vondrick *et al.* [13] also found that an effective object tracker can be obtained by training on a frame-colorization task. In fact, a recent study of the relationships between a plethora of visual tasks explicitly highlights the non-triviality of the relationships between some of them [14].

Hierarchical Priors: One prominent use of auxiliary classifiers, beyond the scope of MTL, is representing hierarchical knowledge. As the categories of classification problems are often semantically organized this way (*e.g.*, objects, places, animals, species, breeds), some work has focused on the benefits of encoding said priors (probabilistic and deterministic). The hierarchical relations can be known a-priori and used for adjusting a prediction by modeling relations of exclusion or subsumption [15]. Alternatively, hierarchical relationships can be learned alongside the model to compensate for classes with a small number of samples [16] or a posteriori where labels for a "student" model are represented by the ones learned previously by a "teacher" classifier [17]. Our approach focuses on relationships that can be directly extracted from the available data, including the semantic information contained by the labels. This implies that the relationships are innate (*i.e.*, not learned), thereby not prone to limitations in the models or training schemes.

Regularizing Branches: Another well-known purpose of auxiliary classifiers has been the stabilization of gradient flow for very deep neural networks. Most prominently, auxiliary classifiers were used for training different iterations of the Inception architecture [18], [19]. Here, auxiliary networks were small parallel branches that used the same training objective as the main network. These branches were only used during training and the reported benefits include faster convergence, more stable gradients and regularization. More recently, models constructed via Neural Architecture Search also made use of auxiliary classifiers in a similar fashion [20]. All of these examples use auxiliary classifiers exclusively for training, resort to the exact same loss function, and are not taken into account for prediction.

Heterogeneous Surrogate Constraint: Instead of one auxiliary branch with the same classification objective, benefits have been reported where small binary classifiers are attached at each layer [21]. These binary branches optimize an objective that measures whether features at each layer are relevant for the main prediction *i.e.*, they yielded a true positive or a false negative prediction. This idea has been recently expanded by dynamically attaching branches with a similar classification objective that are trained based on the output distribution of the main classifier [22]. Alternatively, a reconstruction objective imposed to the original supervised cost (instead of the feature relevance score) was shown to improve classification as well [23]. A modern variation of this idea proposes the use of auxiliary classifiers that process corrupted samples that are later used to adjust the weights of the main classifier [24]. Novel work has also explored the benefits of explicitly modeling information about negative classes into an otherwise classic classification problem [25]. Our work also relies on a different auxiliary objective which requires no extra labels. However, we use the notion of grouping which preserves the information of the original labels (the class) better than a notion of feature relevance, negative classes or input reconstruction. This way, the alignment of the classification objective and the SSAL branch provides outputs that are explicit and directly interpretable.

Groups as Auxiliary Prior: the idea of joining classes together has been exploited to improve upon a classification objective. By grouping classes that fall under a more general semantic term (*e.g.*, "*cat*" and "*dog*" are both "*animals*"), a data-augmentation scheme can easily mine additional data samples that relate to the term subsuming the included labels (the super-term) using a search engine [26]. Both the crawled data and the original dataset are passed through a network with two corresponding branches and trained jointly. Note that data for the "auxiliary" task (the branch for super-classes) is disjoint to the one used for the fine-grained task. Moreover,

there is no explicit correspondence between the classification of the super-class and the original class. A different approach starts by assigning the original labels to visually similar groups and training a dedicated feature extractor for each one. At the same time, a soft-gating mechanism is trained to decide which specialized feature extractors should be used, to finally combine their features into one prediction [27]. In this case, there are no auxiliary objectives (grouping is a priori) and there is only one loss with a single prediction per sample.

Instead of a gating mechanism, HD-CNNs [28] utilize a coarse classifier to control a set of specialized branches. Due to the conditional re-routing of samples based on the coarse classifier or the soft-gating, training these models needs to be adaptive and multi-step, the risk of overfitting increases (specialized networks rely on fewer samples), and the computational cost goes up considerably as more fine-grained classifiers are used.

In contrast, we opt for a much simpler setup that is not affected by the number of coarse groups (in terms of computation), beyond the dimensions of the output layer. Our proposed network can be trained jointly and end-to-end using standard optimization algorithms, with no conditional re-routing or special regularization mechanisms.

III. Methods

In this section, we analyze the algorithmic components from SSAL and how they integrate into a traditional classification problem for training and prediction. There are four main components to discuss: grouping criterion, architectural design, training objectives and joint prediction. For each of these components, we introduce emergent hyper-parameters that need to be considered during evaluation.

A. Grouping Criterion

We propose that the autogenous auxiliary objective be based on a grouping of the original classes. Modeling groups explicitly allows a classifier to learn the property of subsumption; a proven useful mean to generate explanations in formal verification systems [29]. To this end, we use a clustering algorithm based on similar principles than the one used by Yan *et al.* [28] but imposing a constrain that ensures balanced clusters. Concretely, given a set of classes $\mathcal{Y}_c = \{y_1, y_2, \dots y_c\}$ we define $\mathcal{Y}_k$ as a partition of $\mathcal{Y}_c$ into k subsets. The grouping starts by constructing a distance matrix $\mathcal{D}_c$, based on the confusion matrix from a pre-trained model: Given a normalized confusion matrix F with the diagonal set to zero, the distance matrix $\mathcal{D}_c$ is constructed by subtracting 1 from it and then making it symmetric by averaging the off-diagonals (Equation 2).

$$\hat{D} = 1 - F \tag{1}$$

$$\mathcal{D}_c = \frac{1}{2}(\hat{D} + \hat{D}^T) \tag{2}$$

Each cluster is initialized with one of the k labels with the highest average distance to all other labels. The next label in $\mathcal{Y}_c$ to be assigned will be the one with the smallest distance to a cluster currently holding less than c/k elements. In case of a tied metric w.r.t. a cluster, a random one among those is used for the assignment. Note that the distance matrix $\mathcal{D}_c$ can be turned into a similarity measure by omitting the inversion of F *i.e.*, skipping Equation 1.

The output of this algorithm is a mapping $\gamma : \mathcal{Y}_c \rightarrow \mathcal{Y}_k$ assigning a single group label to each of the original ground-truth labels. This way, each sample in a labeled dataset is modeled as a triplet $(x_i, y_i, \gamma(y_i))$ representing the input sample, the ground-truth label and the group label it has been assigned to respectively. A more detailed description of the clustering algorithm, can be found in Section A of the supplementary material.

There are two hyper-parameters that we consider for grouping, namely the number of groups to map to, and the criterion used for grouping. While the former is expressed by an integer $2 \leq k \leq C/2$, the latter can prioritize either joining or splitting visually similar ground-truth labels (by controlling how $\hat{D}$ is computed).

B. SSAL Architectural Design

The proposed model follows the structure of a hard parameter sharing architecture for MTL with three main components, as shown in Figure 1. First, an initial, shared branch h is in charge of extracting low-level features. Next, these common features are fed into two branches f and g with different classification objectives: one with the original ground-truth classification objective, while the second branch optimizes over the group labels. Given an input sample, the ensemble model will output a prediction for the original classification target $f(x)$ and a prediction for the auxiliary task based on grouping $g(x)$.

In practice, these architectures are realized by taking a traditional classifier like Resnet50 [30], and attaching an auxiliary classifier (with a group objective) at some point in-between the layers of the original model. The specific layer disposition for both auxiliary and original models depends on the experiment but in essence, a mixture of convolutional and pooling layers are used. A more detailed specification of all networks used in this work can be found in Section B of the supplementary material.

Under this last perspective, an important hyper-parameter of the architecture is the point at which the auxiliary classifier attaches to the original model. Having a junction in earlier layers allows both branches to work with generic, lower-level features but leaves little room for those features to be regularized by the updates from both branches. Another possible hyper-parameter is the number of auxiliary classifiers that can be attached. In that case, we refer to a set of (possibly different) groupings $\gamma_1, \gamma_2, \dots$ based on the ground-truth labels in $\mathcal{Y}_c$ for which a dedicated auxiliary branch $g_1, g_2, \dots$ is used.

C. SSAL Training

Training relies on traditional end-to-end backpropagation using mini-batch SGD. Both branches f and g are trained

jointly, from scratch, and their individual errors are measured using cross-entropy. Note that there is no unified prediction at this point and the losses for each branch are only added together to force an single update of the entire parameter space, including the common initial feature extractor h. The sum is controlled by weights λ_1 and λ_2 as shown in Equation 3:

$$\mathcal{L} = \lambda_1 \mathcal{L}_f + \lambda_2 \mathcal{L}_g \tag{3}$$

where $\mathcal{L}_f$ and $\mathcal{L}_g$ are the cross-entropy losses for f and g respectively.

D. SSAL Prediction

One of the main novelties of this work is the use of the auxiliary classifier for prediction. To this end, we consider two main alternatives for calculating a joint prediction.

Joint Probability: the final prediction is represented as the joint probability of the original prediction and the auxiliary classifier s.t. $P(y|x) = \mathit{softmax}(f_i(x) \cdot g_{\gamma(i)}(x))$, where f_i is the i-th output dimension of $f(x)$, $g_{\gamma(i)}(x)$ is the output dimension of the auxiliary branch associated with the original label at i and $\cdot$ represents a scalar product. When more than one auxiliary classifier is used, the output of all auxiliary branches g_i is raised to a power $\eta \in (0, 1]$.

Learned Linear Combination: Predictions from $f(x)$ and $g(x)$ are concatenated and then used to train a linear classifier with the same number of outputs as there are labels in the original ground-truth. Both f and g are assumed to be trained already, and the linear classifier is hence trained separately.

We also consider a baseline where only the prediction of $f(x)$ is evaluated. This way, we can establish the influence that jointly training the auxiliary classifier has had in the performance of the branch with the original classification problem. In other words, this baseline evaluation measures the inductive bias of the auxiliary classifier.

IV. Experiments

In this section we describe the datasets, hyper-parameters, baselines and performance experiments to support and quantify the benefits of SSAL models.

A. Datasets

We conduct experiments on three different image classification datasets with varying degrees of complexity:

CIFAR100 [5]: extension of CIFAR10 where 60 000 color images of size 32x32 belong to 100 different classes of fine-grained objects or animals. The training and test set contain 50 000 and 10 000 images respectively.

TinyImagenet [6]: 110 000 color images of size 64x64 split into 200 natural categories *e.g.*, animals, food, furniture. They are divided into 100 000 samples for training and 10 000 for validation. The official testing set does not provide labels, hence we take a small portion of the training set for development and report results on the validation set.

Imagenet [7]: One of the largest image classification datasets available. Image size is variable but samples are commonly downscaled to 300x300 pixels. They comprise over 1.2M images across 1000 categories. Similarly to TinyImagenet, we use the 50 000 validation samples for testing and in turn, take a small portion of the training set for any validation that is required.

B. SSAL Hyper-Parameters

As mentioned in Section III, SSAL models introduce a variety of hyper-parameters requiring additional consideration.

Layer Architecture: as mentioned earlier, multiple architectures are used depending on the dataset and the objective of the experiment. We base our evaluations and SSAL models on five high-performance architectures: Resnet18 [30], Resnet50 [30], Wide-Residual-Networks (WRNs) [31], Squeeze and Excitation Nets (SENets) [32], and DenseNets [33]. For the architecture of auxiliary branches we use a combination of blocks comprising convolutional, pooling, batch-normalization and inception-like layers. When a hyper-parameter search on these architectural elements is required, we use a small portion of the training set for validation, before evaluating on the corresponding test set.

Grouping Criterion and Number of Groups: we train SSAL models based on Resnet18 for CIFAR100 and TinyImagenet. The auxiliary classifier consists of four convolutional layers with batch-normalization and ReLU activation, a global average pooling, two fully connected layers and a final linear combination with softmax normalization. The size and number of convolutional filters, and the number of fully connected neurons were determined via hyper-parameter search. See Section C of the supplementary material for further details about said parameters.

We use a single auxiliary classifier (a.k.a. SSAL branch) with either 2, 4, 10 or 20 groups, and a grouping criterion that either splits or joins visually similar classes following the computation outlined in Section III-A. The model prediction is done by computing the joint probability as proposed in Section III-D. The auxiliary classifier attaches to the main network between the first max-pooling and the first residual block.

Results in Figure 2 show a constant improvement of the combined classification error as the number of groups increases. Although grouping visually similar classes yields an initial small advantage compared to the "splitting" criterion, this tendency inverts when the number of groups reaches 10 and 20. The pattern, albeit some marginal fluctuations, is preserved for both datasets.

This first experiment suggests that having more groups is beneficial and that either splitting or joining visually similar classes contribute to a better performance at a similar rate.

Position of the Auxiliary Classifier: we use a similar setup based on Resnet50 for CIFAR100 and vary the point at which the auxiliary classifier is attached. Each of the four residual blocks in the original network is considered an atomic unit. We evaluate the effects of attaching the auxiliary classifier after each one of said blocks. The architecture of the auxiliary classifiers remain the same, except for the number of channels in the first layer which increases as the point of attachment

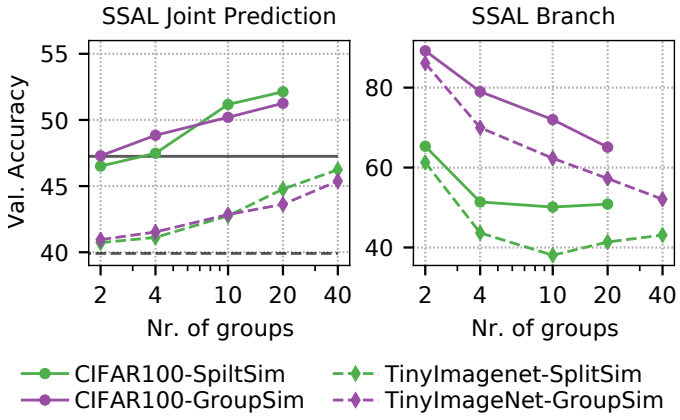

Fig. 2. Influence of the number of groups and grouping criterion on the accuracy of CIFAR100.

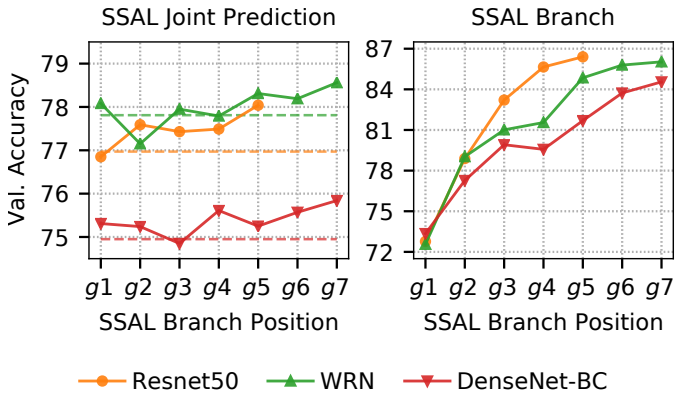

Fig. 3. Accuracy when the position of the auxiliary classifier (with 20 visually similar groups) varies w.r.t. the main network.

lies deeper in the original network. They all optimize over the same 20 groups joining visually similar classes, and the final prediction is done via the joint probability. The same experiment is conducted using two different classifiers: a Wide Residual Network (WRN 28-10) and a DenseNet (DenseNet-BC 100-12). Seven points of attachment at different depths are selected for each network. These points include paths that lie before, after and in-between macro-blocks (see Figure 4).

Results in Figure 3 show that the position of the auxiliary classifier w.r.t. the main model has a tendency to perform best when the auxiliary classifier is attached at deeper layers in the original network. This behaviour corresponds with the performance of the SSAL branch itself, which shows higher performance when it has been attached at a deeper stage within the architecture.

Number of Auxiliary Classifiers: to evaluate the influence of attaching more than one auxiliary classifier to the main model, we train Resnet18 on CIFAR100 while either one or two auxiliary classifiers g_1, g_2 are attached. The auxiliary classifiers are both composed of two convolutional layers with batch-normalization and ReLU followed by an inception-like layer, global average pooling, and a linear output layer with

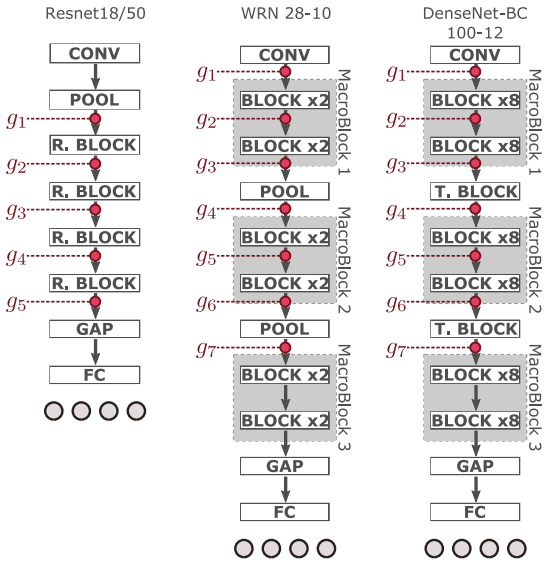

Fig. 4. Positions where SSAL branches are being attached to the main network (red circles). Only one auxiliary branch is evaluated at a time.

softmax normalization. The auxiliary branch g_1 is placed after the first residual block and optimizes over 20 visually similar groups. The g_2 counterpart is placed after the second residual block, and optimizes over 50 visually similar groups. Final prediction is based on the joint probability, and an equal normalization power $\eta = 1$ is used.

Two similar experiments are conducted using Resnet50 and WRN. For these two variants, three auxiliary branches are simultaneously attached. Grouping is based on visual similarity and they optimize over an increasing number of groups: 20, 30 and 50 groups. The normalization power is the same for all branches. Results are summarized in Table I.

Increasing the number of SSAL branches does have a positive impact on performance, as long as the normalization power η decreases when the number of SSAL branches goes up. Intuitively, the role of SSAL branches is one of verification and support rather than a predominant signal, and thereby

	$\|\|g_i\|\|$	η	Test Acc. (%)	Parameters
Resnet18	0	-	75.67	11.23M
	1	1.0	76.62	11.92M
	2	1.0	**78.23**	12.83M
Resnet50	0	-	79.13	23.77M
	1	1.0	79.70	25.07M
	3	1.0	80.36	28.89M
	3	0.3	**80.69**	28.89M
WRN 28-10	0	-	80.19	36.56M
	1	1.0	80.96	38.19M
	3	1.0	80.68	43.25M
	3	0.4	**81.08**	43.25M

TABLE I
CLASSIFICATION ACCURACY FOR MODELS WITH $||g_i||$ SSAL BRANCHES ON CIFAR100. USING MORE BRANCHES, TOGETHER WITH REGULARIZATION, IMPROVES PERFORMANCE. HOWEVER THE COMPUTATIONAL FOOTPRINT ALSO INCREASES.

outputs from these branches should be weighted down in scenarios where branches outnumber the original classification network. For Resnet18, adding two SSAL branches yields an accuracy of 78.2%, a 2.6 pp. improvement over the baseline.

C. Alternative Baselines

The use of auxiliary branches inevitably adds more raw capacity to the overall network by virtue of the extra trainable parameters. We test whether the consistent boost in performance can be simply explained by the additional weights (Occam's razor) or if the introduction of the SSAL objective has merit on its own.

To this end, we train modified versions of Resnet18 on Tiny-Imagenet that add more weights in various ways, matching or surpassing the number of parameters of a SSAL model. We also compare models with the same architectural design of SSAL models but training without the SSAL objective:

- **WideResnet18**: has 50% more filters across all convolutional layers.
- **DeepResnet18**: adds four convolutional layers of 256 filters each with batch-normalization and ReLU activations before the first residual block.
- **DWResnet18**: similar to DeepResnet18 but doubling the number of filters of the additional convolutional layers.
- **GapCatNoSSAL**: based on a SSAL model but without the SSAL loss. The output of the SSAL branch is concatenated with the GAP activation of the main classifier.
- **CatFCNoSSAL**: based on the SSAL architecture but without the SSAL objective. The outputs of the SSAL branch and the original network are concatenated and passed through a fully-connected layer with 2048 neurons. The result is then passed through a linear combination for the final prediction.
- **LinearComb**: as described in Section III-D we start with a fully trained SSAL model, but instead of issuing predictions through a joint probability, both the auxiliary output and the prediction from the original classifier are concatenated together and used to train a separate linear classifier.
- **SSAL**: classifier ensemble proposed in this work. For the variant with one auxiliary classifier, the SSAL branch is placed after the first residual block ($g2$ in Figure 4) while the model with three SSAL branches uses attachment points corresponding to $g1, g2$, and $g3$.

Networks are trained for 20 epochs with a triangular learning rate peaking at epoch 8. Each experiment is repeated three times to account for initialization effects. Results are summarized in Table II.

It is clear that adding more capacity to Resnet18 improves accuracy. Capacity in the form of deeper layers shows better results than using wider layers, and a combination of both yields an overall improvement of up to 3.8 percentage points. Using the same architectural disposition of a SSAL model but without the SSAL objective (*NoSSAL), worsen performance w.r.t. the baseline, discarding this setup as the reason for improvements. Overall, the use of SSAL objectives remains the most effective use of the extra weights and layers with an improvement over the baseline of 5.9 to 12.7 percentage points (8.9 points better than the best baseline, while keeping a lower parameter count).

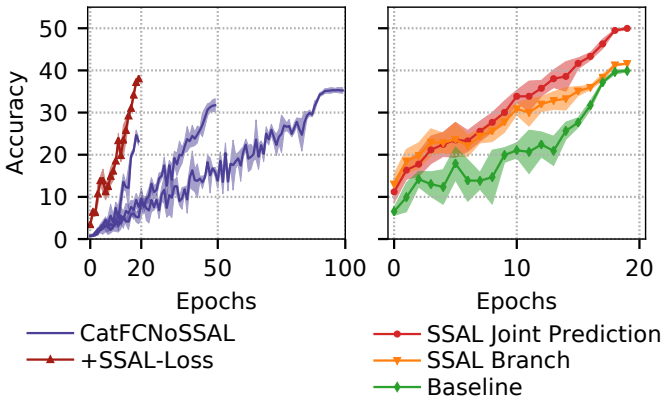

Fig. 5. Accuracy of a SSAL model based on Resnet18 (red) is not only higher, but also reached with less training than the same model without the SSAL objective (purple) or even the original Resnet18 (green).

Training Convergence: we measure the rate of convergence when training a SSAL model on CIFAR100, verifying that the SSAL objective has an aligned inductive bias which is not only beneficial for classification but it also requires less training steps. We train the *CatFCNoSSAL* baseline for 20, 50 and 100 epochs, and compare it with a fourth run where the SSAL objective is added to the training procedure (the model architecture remains identical). Results in Figure 5 (left) show the validation accuracy of these four systems. Here we see that training with the SSAL objective drastically accelerates convergence. Even after 100 epochs, an identical architecture is still unable to match the performance of its SSAL counterpart. Effects of this accelerated convergence is also evident when comparing the training accuracy of a SSAL-based pipeline (joint prediction) against a baseline implementation with no auxiliary branches (Figure 5 right).

D. Improving Classification

Based on the analysis of hyper-parameters for SSAL models, we show that high accuracy is consistently attainable

	Val. Acc. (%)	Diff (pp.)	Parameters
Resnet18	39.9 ± 0.3	0.0	11.2M
WideResnet18	42.3 ± 0.3	2.4	25.3M
DeepResnet18	43.1 ± 0.4	3.2	13.3M
DWResnet18	43.7 ± 0.1	3.8	19.0M
GapCatNoSSAL	40.2 ± 0.3	0.3	15.6M
CatFCNoSSAL	35.3 ± 0.8	-4.6	13.6M
LinearComb	**44.1 ± 0.1**	**4.2**	12.8M
SSAL x1	**45.8 ± 0.2**	**5.9**	12.6M
SSAL x3	**50.0 ± 0.4**	**10.1**	15.6M

TABLE II
BASELINES FOR SSAL MODELS ON TINYIMAGENET. MODELS THAT HAVE A DEEPER ARCHITECTURE, WIDER LAYERS, OR LACK THE SSAL OBJECTIVE FAIL TO REACH THE LEVEL OF ACCURACY OF SSAL MODELS.

across a variety of well-known, thoroughly optimized architectures.

CIFAR100: We train SSAL models based on Resnet50, WRN, SENet and DenseNet on CIFAR100. For each of these original architectures, we attach three SSAL branches with visually similar groups of 20, 33, and 50 groups. To guarantee uniformity on the evaluation conditions, we have re-implemented all models and trained them from scratch so that the only difference between the original performance and the SSAL variant is the proposed surrogate objective. Additionally, we include baselines from their original source (org), our own re-implementation (ours), and the *LinearComb* setup from Section IV-C (+LC). For SSAL models, we report the accuracy of the original classifier *i.e.*, using the SSAL branch during training but not for prediction (+TR), and the full SSAL prediction using the joint probability (+JP). For further details about the architecture of the SSAL branches and the training setup, please refer to Section C in the supplementary material. Table III summarizes the results.

These experiments show that training with the auxiliary classifier consistently yields better performance. The inductive bias of SSAL branches guides the classifier even when the auxiliary output is not used for prediction. Performance improves even further when SSAL models issue a joint prediction. Note that for WRN and DenseNet, the SSAL version outperforms the original model (org), notwithstanding the weaker baseline it starts from (ours).

Imagenet: To test the effects of SSAL branches on large scale problems, we train a Resnet50 on Imagenet (ours), and compare it with a corresponding SSAL model with three auxiliary branches. As in the previous experiment, they use visually similar classes with 200, 334, and 500 groups, and report values for training with SSAL only (+TR), joint prediction (+JP) and using the *LinearComb* setup from IV-C (+LC). We also evaluate on a *GapCatNoSSAL* baseline (GC) from IV-C which has a similar architecture but no SSAL objective. Table IV shows how, once again, a SSAL model is able to outperform the original baseline by almost 1.5 p.p..

Finally, in Table V, we compare our results with recently proposed state-of-the-art methods described in Section II. In particular, we show that SSAL models outperform alternative methods conveying contextual information in the loss function, use other kind of auxiliary classifiers or rely on different hierarchical priors for training.

	Val. Accuracy					Parameters	
	org	ours	+TR	+JP	+LC	org	SSAL
Resnet50	-	78.94	79.67	80.61	80.23	23.8M	28.9M
SE-WRN 16-8	80.86	79.02	79.02	80.20	80.03	11.1M	14.9M
WRN 28-10	80.75	80.08	80.65	**80.97**	80.68	36.6M	38.2M
DenseNet 190-40	82.82	81.06	81.85	**83.24**	83.10	26.1M	38.3M

TABLE III
CLASSIFICATION ACCURACY FOR MULTIPLE HIGH-PERFORMANCE ARCHITECTURES ON CIFAR100. ADDING THE SSAL OBJECTIVE CONSISTENTLY YIELDS HIGHER PERFORMANCE.

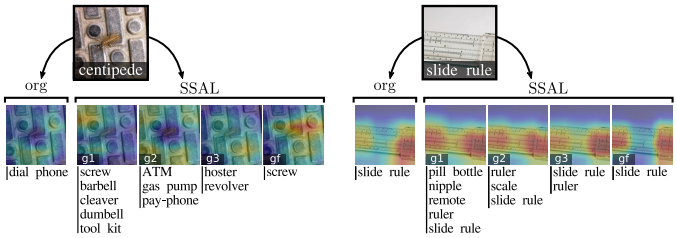

Fig. 6. CAM w.r.t. each auxiliary branch g_i of the SSAL model. g_f denotes the final classification output of the SSAL model, and **org** is the CAM of a normal Resnet50. Labels within predicted SSAL groups are shown below each branch.

E. Contextual Validation

We show that predictions of SSAL models are more interpretable than regular DNNs thanks to the grouping objective of their auxiliary branches. The use of heatmaps has been controversial as a mean to interpret a model's output because it can only point to the area of importance while leaving out information about the underlying features that elicit a high response [3]. Labels within each SSAL group can be used to identify which low-level features are responsible for the prediction.

Figure 6 shows the Class Activation Mapping [34] of two examples: a false-, and a true-positive. For the former, predicted SSAL groups contain labels with metallic parts, and box-like shapes which correspond to areas with a strong activation. For the slide rule, class labels in auxiliary groups like "pill bottle" or "nipple" (mouthpiece of a baby bottle) often depict the uniform markings found in rulers; a strong indication that these are precisely the salient features that guided this particular prediction.

In contrast, regular classifiers provide less nuanced insights where multiple interpretations are possible. Examples in Figure 6 leave ample room for interpretation when predicting "dial phone" or even the true positive for "slide rule" (`org`). More

	ours (org)	+TR	+JP	+LC	GC
Top-1	75.5 ± 0.1	76.4 ± 0.1	$\mathbf{76.9 \pm 0.1}$	76.6 ± 0.2	75.7 ± 0.1
Top-5	92.7 ± 0.1	93.3 ± 0.1	$\mathbf{93.7 \pm 0.1}$	93.4 ± 0.1	92.7 ± 0.1

TABLE IV
ACCURACY FOR SSAL RESNET50 ON IMAGENET. EXPERIMENTS ARE RUN 3 TIMES.

	CIFAR100	Imagenet
HD-CNN [28]	65.64	68.66 (-)*
HydraNets [27]	76.25	73.20 (-)*
COT [25]	79.46	75.60 (-)
DSL [21], [22]	81.95	76.12 (92.93)
DHM [22]	82.80	76.57 (93.24)
Aux. Train [24]	80.84	74.14 (-)*
SSAL (ours)	**83.24**	**77.00 (93.80)**

TABLE V
TOP-1 ACCURACY OF RELATED STATE-OF-THE-ART AND SSAL MODELS. RESULTS FOR IMAGENET ARE BASED ON RESNET50 EXCEPT THE ONES MARKED WITH *. TOP-5 SHOWN IN PARENTHESIS, IF AVAILABLE.

examples can be found in the supplementary material.

V. Conclusions

In this work, we have introduced SSAL: a method for extending neural network architectures with auxiliary objectives that are related to the original supervised task. These objectives express low-level priors (*e.g.*, grouping) which do not require any additional manual annotation, but are derived from a pre-existing annotated set *i.e.*, they are *autogenous*. SSAL models follow the structure of multi-task learning algorithms, making a joint prediction based on the outputs from all its branches possible. We show that the use of SSAL objectives consistently yields higher classification performance across several state-of-the-art classifiers like Resnets, DenseNets, SENets and WRNs for different datasets like CIFAR100, TinyImagenet and Imagenet. The usefulness of the SSAL objective is validated through a comparison with several baselines including networks with similar architectural structure but no SSAL objective and networks with a comparable number of parameters. Finally, we show that SSAL models capture more evidence for each sample, enabling a more grounded interpretation of predictions.

Future Work: we are interested in measuring the effects of different clustering algorithms that can be used as grouping criterion (*e.g.*, bootstraping or EM-like strategies) Furthermore, we want to explore other low-level priors beyond grouping that can be conveyed in a self-supervised fashion. Finally, we are interested in assessing the benefits of using SSAL branches for neural-architecture-search algorithms.

VI. Acknowledgments

This work was supported by the BMBF projects ExplAINN (01IS19074), DeFuseNN (01IW17002) and the NVIDIA AI Lab program.

References

[1] A. Bendale and T. Boult, "Towards open world recognition," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[2] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," *arXiv preprint arXiv:1702.08608*, 2017.

[3] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nature Machine Intelligence*, 2019.

[4] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *International Conference on Learning Representations*, 2014.

[5] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[6] F.-F. Li, A. Karpathy, and J. Johnson. (2017) Tiny imagenet. [Online]. Available: https://www.kaggle.com/c/tiny-imagenet/

[7] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, 2015.

[8] Y. S. Abu-Mostafa, "Learning from hints in neural networks," *Journal of complexity*, vol. 6, no. 2, 1990.

[9] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*, D. E. Rumelhart, J. L. McClelland, and C. PDP Research Group, Eds. Cambridge, MA, USA: MIT Press, 1986, ch. Learning Internal Representations by Error Propagation, pp. 318–362.

[10] R. Caruana, "Multitask learning," *Machine learning*, vol. 28, no. 1, 1997.

[11] L. Kaiser, A. N. Gomez, N. Shazeer, A. Vaswani, N. Parmar, L. Jones, and J. Uszkoreit, "One model to learn them all," *arXiv preprint arXiv:1706.05137*, 2017.

[12] H.-Y. Lee, J.-B. Huang, M. Singh, and M.-H. Yang, "Unsupervised representation learning by sorting sequences," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[13] C. Vondrick, A. Shrivastava, A. Fathi, S. Guadarrama, and K. Murphy, "Tracking emerges by colorizing videos," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[14] A. R. Zamir, A. Sax, W. Shen, L. J. Guibas, J. Malik, and S. Savarese, "Taskonomy: Disentangling task transfer learning," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[15] J. Deng, N. Ding, Y. Jia, A. Frome, K. Murphy, S. Bengio, Y. Li, H. Neven, and H. Adam, "Large-scale object classification using label relation graphs," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014.

[16] N. Srivastava and R. R. Salakhutdinov, "Discriminative transfer learning with tree-based priors," in *Advances in Neural Information Processing Systems*, 2013.

[17] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[18] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[19] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[20] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[21] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, "Deeply-supervised nets," in *Artificial Intelligence and Statistics*, 2015.

[22] D. Li and Q. Chen, "Dynamic hierarchical mimicking towards consistent optimization objectives," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[23] Y. Zhang, K. Lee, and H. Lee, "Augmenting supervised neural networks with unsupervised objectives for large-scale image classification," in *International Conference on Machine Learning*, 2016.

[24] L. Zhang, M. Yu, T. Chen, Z. Shi, C. Bao, and K. Ma, "Auxiliary training: Towards accurate and robust models," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[25] H.-Y. Chen, P.-H. Wang, C.-H. Liu, S.-C. Chang, J.-Y. Pan, Y.-T. Chen, W. Wei, and D.-C. Juan, "Complement objective training," in *International Conference on Learning Representations*, 2019.

[26] S. Xie, T. Yang, X. Wang, and Y. Lin, "Hyper-class augmented and regularized deep learning for fine-grained image classification," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[27] R. Teja Mullapudi, W. R. Mark, N. Shazeer, and K. Fatahalian, "Hydranets: Specialized dynamic architectures for efficient inference," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[28] Z. Yan, H. Zhang, R. Piramuthu, V. Jagadeesh, D. DeCoste, W. Di, and Y. Yu, "Hd-cnn: hierarchical deep convolutional neural networks for large scale visual recognition," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

[29] D. L. McGuinness and A. Borgida, "Explaining subsumption in description logics," in *IJCAI (1)*, 1995.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[31] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.

[32] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[33] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[34] B. Zhou, A. Khosla, L. A., A. Oliva, and A. Torralba, "Learning Deep Features for Discriminative Localization." *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.