The Visual Computer (2021) 37:1593–1610
https://doi.org/10.1007/s00371-020-01922-5

ORIGINAL ARTICLE

# Efficient image dataset classification difficulty estimation for predicting deep-learning accuracy

Florian Scheidegger[1,2] · Roxana Istrate[2,3] · Giovanni Mariani[2] · Luca Benini[1,4] · Costas Bekas[2] · Cristiano Malossi[2]

Published online: 28 July 2020


**Abstract**
In the deep-learning community, new algorithms are published at a very fast pace. Therefore, solving an image classification problem for new datasets becomes a challenging task, as it requires to re-evaluate published algorithms and their different configurations in order to find a close to optimal classifier. To facilitate this process, before biasing our decision toward a class of neural networks or running an expensive search over the network space, we propose to estimate the classification difficulty of the dataset. Our method computes a single number that characterizes the dataset difficulty 97× faster than training state-of-the-art networks. The proposed method can be used in combination with network topology and hyper-parameter search optimizers to efficiently drive the search toward promising neural network configurations.

**Keywords** Dataset characterization · Classification difficulty · Deep learning · Image classification

## 1 Introduction

Convolutional neural networks (CNNs) gained popularity in recent years thanks to the availability of powerful GPUs that enable to efficiently train accurate classification models [19]. Visual deep learning enables applications like 3D shape classification [32], multi-label image classification [6], Flyer classification [39], face detection [7], and automatic car counting [35]. For building practical applications, the deep-learning community shares a common interest in reducing the development cycle, while increasing model accuracy and keeping infrastructure and power consumption expenditure under control. Many publications address these conflicting goals [10], [17], [18]. Most machine-learning approaches require a human in the loop responsible for taking crucial decisions such as defining the network, finding good combinations of hyper-parameters and performing adequate preprocessing on the input data. To overcome the problem of manual selection, various automated approaches such as grid search, random search [3], Bayesian optimization [46] or hyperband optimization [30] have been proposed. These methods operate autonomously and improve model performance; however, they still suffer from two limiting factors. First, they require a definition of the search space. Second, they consume a large amount of resources for a single optimization task.



✉ Florian Scheidegger
eid@zurich.ibm.com

Roxana Istrate
roi@zurich.ibm.com

Giovanni Mariani
ova@zurich.ibm.com

Luca Benini
lbenini@iis.ee.ethz.ch

Costas Bekas
bek@zurich.ibm.com

Cristiano Malossi
acm@zurich.ibm.com

1 ETH Zürich, Rämistrasse 101, 8092 Zürich, Switzerland

2 IBM Research - Zürich, Säumerstrasse 4, 8803 Rüschlikon, Switzerland

3 Queen's University of Belfast, University Road, Belfast, BT7 1NN Northern Ireland, UK

4 Università di Bologna, Via Zamboni 33, 40126 Bologna, Italy

In this paper, we propose automated methods for quantifying the *difficulty* of a classification problem in terms of how hard it is to reach high accuracy for a given dataset. Our developed neural network architectures, the probe nets, reliably forecast how well any machine learning will perform on the given dataset. We designed probe nets to be small to allow fast training, and still, they provide a scoring value that consistently outperforms the three alternative scoring pipelines. The proposed probe nets can be used in combination with architecture search optimizers to efficiently drive the search toward promising configurations, avoiding the exploration of unsuitable networks. Consciously or not, the characterization of dataset difficulty is a process, followed by every deep learning architect. When looking for a well-performing model for a new dataset, common practice is to try state-of-the-art networks to evaluate how hard is to classify the images in the dataset. Since datasets are large and models complex, the process of training, comparing, and selecting a few state-of-the-art deep networks becomes a computationally heavy task. Probe nets improve this step by providing a classification difficulty estimator, which provides insights into the classification task and can be used to rapidly confine the exploration to a few promising networks.Our developed probe nets characterize datasets orders of magnitude faster than the actual training and have high correlation with state-of-the-art network accuracies.

In summary, our main contributions are the following:

- We conduct a large literature and experimental study on reference machine learning algorithms, including 484 cited state-of-the-art results.
- We reproduced results of thirty deep learning models over sixteen datasets to defend the key observation that no single algorithm outperforms the alternatives. We establish the reference dataset classification difficulty on the ensemble of reference results.
- We propose and evaluate four dataset complexity scoring pipelines to estimate the classification difficulty.
- We develop probe nets as suitable candidates to efficiently and reliably estimate the classification difficulty.
- We evaluate approximate computing techniques, such as subsampling and early stopping, to further reduce the execution time without affecting the final results.
- We showcase the proposed dataset characterization used in an architecture search setting.

The remainder of the paper is organized as follows: Section 2 describes the related work, and Sect. 3 introduces the notation and presents an overview of reference models and their performance on datasets. Section 4 details the adopted methodologies. Section 5 examines the results. Section 6 studies efficient implementations, Sect. 7 demonstrates how the methodologies are applied to perform an efficient architecture search, and Sect. 8 concludes all findings.

## 2 Related work

The topic of difficulty estimation of a dataset is scarcely explored in the literature. In [51], the authors address the difficulty of visual search within one image by assessing the human response time for solving a visual search task. Their technique employs two VGG-like [45] networks that work as encoders and extract features that are further passed through a regressor to produce a per image score. In contrast, our technique focuses on defining how easily separable are different classes within a dataset; henceforth, our proposed score measures how challenging a classification instance is. We omit a direct comparison since their reference consists of measured human response times including high variances causing their approach to produce modest correlations. In contrast, we motivate and validate our score based on a extensive set of reference results, causing stable results and strong correlations in our best proposed approach.

In [22], the authors propose measures that characterize the difficulty of a classification problem, focusing on the geometrical complexity of the class boundary. However, those measures are defined and evaluated over binary classification problems defined over a low-dimensional feature vector space. In contrast, we focus on image classification problems building a high-dimensional spatially correlated data space consisting of multiple classes.

The $H$-divergence is a rigorous measure computed between two datasets in the field of domain adaption [2,15,28] where a source and target distribution of datasets are compared. In contrast, our approach in this work scores the classification complexity stemming from a single dataset. Scoring datasets independent of additional information allows to efficiently extend a collection of datasets and their scores without computing cross-interactions between multiple datasets.

The latest research on neural network design and network architecture search is accounted for by considering suggested architectures in this work. We reproduce all results for a fair comparison. That approach covers variants from residual bypass operations (ResNets [20]) to even high fan-out and convergent structures such as they occur in the inception module [48] or DenseNets [24]. Structures that favor lighter operations for better performance on constraint devices, such as MobileNets [23], and by architecture search generated structures, such as PNASNets [31], are included in our study.

The main concepts and key ideas of characterizing datasets are available in a preprint [44]. This work summarizes those findings and extends them in three ways: First, the established reference dataset characterization is enlarged from one reference architecture to an ensemble of state-of-the-art

architectures; second, a novel and efficient Fréchet inception distance-based pipeline is added, and third, a use case of architecture search demonstrates time critical design choices and the usefulness of the dataset characterization in a real application.

## 3 Datasets and reference models

### 3.1 Notation

In this work, we refer to a dataset with the quadruple $\mathcal{D} := (\mathbf{X}_{train}, \mathbf{y}_{train}, \mathbf{X}_{test}, \mathbf{y}_{test})$, where $\mathbf{X}_{train} \in \mathbb{R}^{n_{train} \times d}$ and $\mathbf{X}_{test} \in \mathbb{R}^{n_{test} \times d}$ are the training and testing inputs, and $\mathbf{y}_{train} \in [1, C]^{n_{train}}$ and $\mathbf{y}_{test} \in [1, C]^{n_{test}}$ are the training and testing labels. We assume that the datasets come already split into train and test sets, as this is commonly the case for published data. We denote the dimension of the input samples as $d$, the number of input samples as $n_{train}$ for training and $n_{test}$ for testing, and the number of classes as $C$. $\mathcal{M}$ refers to a model, including the network topology and related hyper-parameters, and it includes the training and data augmentation-related hyper-parameters. Therefore, the tuple $(\mathcal{D}, \mathcal{M})$ specifies a deep learning training run of model $\mathcal{M}$ on dataset $\mathcal{D}$. We denote with Top-1$(\mathcal{D}, \mathcal{M})$ the Top-1 accuracy classification performance of the training run. In all experiments, training is performed with $(\mathbf{X}_{train}, \mathbf{y}_{train})$ and performance is measured on $(\mathbf{X}_{test}, \mathbf{y}_{test})$.

### 3.2 Datasets and literature referenced results

In this work, we focus on sixteen public available and established image classification datasets as presented in Table 1. Figures 1, 2 depicts the number of samples used for training and testing. The datasets span two order of magnitudes in the number of classes and in the number of available training samples and one order of magnitude in the balance ratio. The datasets stem from various domains and cover typical and relevant use cases such as optical digit recognition stemming from handwritten samples (*MNIST*) or in the context of images stemming from house numbers (*svhn*). *GTSRB* covers traffic sign recognition, a use case that occurs in autonomous driving systems. Scene recognition aims to classify the location of where the picture was taken as whole (*indoor67* and *places*), whereas traditional classification tasks are posed around identifying a class based on a particular object present within the image. Figure 1 shows external generated and collected state-of-the-art results. Various known machine learning solutions spread over a large performance range, except for simple datasets, such as *MNIST* [13], where most methods perform well. Highlighted is the best experimental value we reproduced in a controlled environment as follows in Sect. 3.3. These

**Table 1** Dataset overview

| Dataset | #classes | Balance† | Test split††(%) |
|---|---|---|---|
| *flowers102* [37] | 102 | 1.00 | 75.1 |
| *flowers*[1] | 5 | 1.50 | 13.6 |
| *textures* [8] | 47 | 1.00 | 33.3 |
| *stl10* [9] | 10 | 1.00 | 61.5 |
| *indoor67* [40] | 67 | 1.08 | 20.0 |
| *caltech256* [16] | 257 | 11.91 | 33.0 |
| *GTSRB* [47] | 43 | 10.71 | 24.4 |
| *CIFAR10* [27] | 10 | 1.00 | 16.7 |
| *CIFAR100* [27] | 100 | 1.00 | 16.7 |
| *MNIST* [13] | 10 | 1.24 | 14.3 |
| *fashion MNIST* [54] | 10 | 1.00 | 14.3 |
| *food101* [4] | 101 | 1.00 | 25.0 |
| *quickdraw*[2] | 345 | 1.00 | 9.1 |
| *places* [56] | 84 | 1.45 | 2.0 |
| *svhn* [36] | 10 | 1.67 | 4.1 |

The number of classes per dataset covers two orders of magnitude, from as few as 5 classes up to 345 classes. The balance ratio spans from 1.0 (for equally balanced datasets) up to a factor of 11.9×
†ratio of class samples of majority over minority class ††fraction of test samples out of all samples

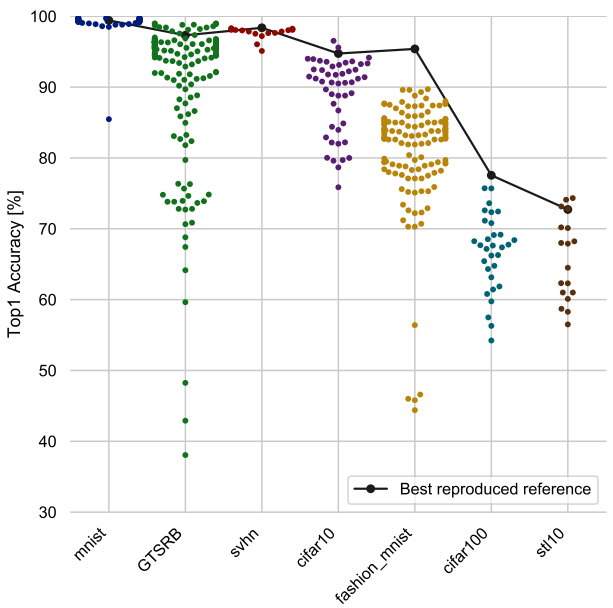


**Fig. 1** State-of-the-art results achieved on datasets with more than twenty reference results. Each point corresponds to a single published algorithm, capturing vanilla CNNs with or without transfer learning, non-deep learning approaches such as SVMs or random forests applied on handcrafted features, and other problem specific pipelines. Highlighted in black is the best reference solution we reproduced in a controlled environment

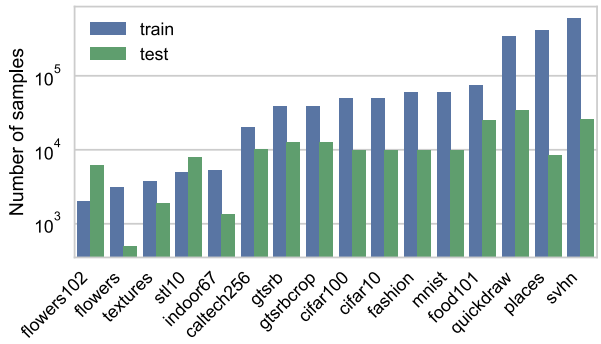


**Fig. 2** The number of samples within a given dataset used for training and testing sorted by training samples. Train and test sets are always disjoint, and the splitting is given as suggested by the reference. The number of training samples spans more than two orders of magnitude

accuracy results are competitive with top performing published results and outperform published solutions in the case of *fashion MNIST* since the referenced results stem from an extensive study of traditional machine learning approaches without including deep learning algorithms [54].

### 3.3 Reference models

In order to evaluate the reference difficulty of an image classification problem, we report the accuracy obtained by an ensemble of established deep learning network topology. Since there exists no single network model that performs best on all available datasets nor are reference accuracies for all models published on all considered datasets, we reproduced reference accuracies for reference models within this work with an extensive set of experiments. We use a predefined list of well-established network architectures and report achieved accuracies on the dataset considered. Table 2 summarizes the network models we consider as established reference models. Most of them are provided with family-specific and topology-related hyper-parameters, such as, for example, VGG or ResNets where the parameter refers to the total amount of layers and controls the overall complexity of the model. All models are well established, and the only modification we performed was to adapt the weight matrix of the last output layer such that the number of output neurons matches the number of classes for the given dataset. Figure 3 shows averaged normalized execution times for one batch of size 128 for training and testing. All experiments in this paper are obtained with PyTorch version 0.4.1 and run on an IBM Power8 equipped with a P100 GPU. Timings are measured in a realistic setting, e.g., including occurring overheads of loading and transferring data between CPU and GPU and kernel lunch overheads. Training times of one batch include operations caused by back propagation and the weight update, while testing times refer to the elapsed time the model required to perform a batched forward inference. The fastest model considered is *LeNet* that trains within 30 ms per batch which is 16× faster than *ResNeXt29_4x64d* that takes about 480 ms per training batch.

The next section empirically demonstrates two insights obtained from the ensemble of reference models: First, there does not exist a single model that outperforms all other models on every dataset. Second, for a given datasets, there are trends of how the ensemble of models behaves. The first observation builds a key argument to perform architecture search on new problem instances to find suited models on new datasets. The latter demonstrates the inherent difficulty of the problem instance due to the given classification difficulty present in the data independent of the specific model fitted on it.

**Table 2** Established reference network architectures

| Family | Variant | Max batch size | Instances |
|---|---|---|---|
| ResNet | 18, 34, 50, 101, 152 | 1024–128 | 5 |
| PreActResNet | 18, 34, 50, 101, 152 | 1024-128 | 5 |
| ResNeXt29 | 2×64d, 4×64d, 32×4d | 256–64 | 3 |
| DenseNet | 121, 161, 169, 201 | 256–128 | 4 |
| LeNet | – | 1024 | 1 |
| GoogLeNet | – | 256 | 1 |
| MobileNet | – | 1024 | 1 |
| MobileNetv2 | – | 512 | 1 |
| PNASNet | Type A, Type B | 1024, 512 | 2 |
| DPN | 26, 92 | 512, 128 | 2 |
| SENet18 | – | 1024 | 1 |
| VGG | 11, 13, 16, 19 | 1024 | 4 |
| Total | | | 30 |

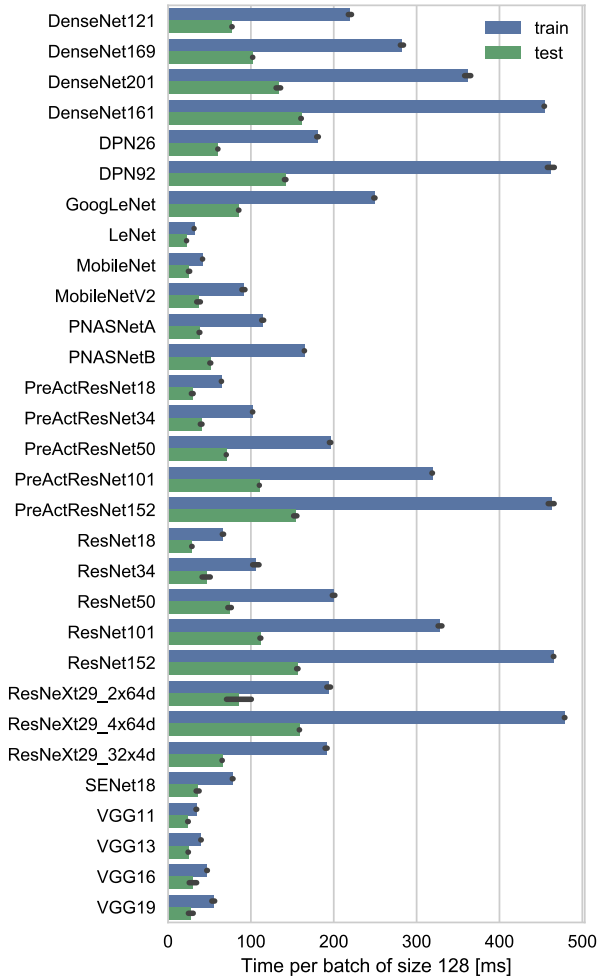


**Fig. 3** Average timings per batch of size 128 for training and testing of reference models. Timings span from 30 ms up to 480 ms per training batch

### 3.4 Reference results as proxy for classification difficulty

Figure 4 shows all reference accuracies obtained when training established models on the considered datasets as described in Sect. 3. Results of the models cluster at a dataset-specific accuracy saturation. That saturation limit demonstrates the existence of a data inherent classification difficulty for machine learning algorithms independent of the model choice. We empirically demonstrate that a single best model does not exist by highlighting some of the results presented in Fig. 4. Table 3 reports which reference model has achieved the best accuracy and henceforth represents the accuracy saturation limit of current state of the art. Out of the 16 problem instances imposed by the 16 datasets, twelve different reference models belong to the best performer and some of them were multiple times selected as best candidate. However, none of the reference architecture is able to clearly outperform the others. In some cases, such as for *MNIST*, all reference accuracies obtained by various models are close which makes it challenging or impossible to distinguish two different models or to select the best model. To address this fact, Table 4 demonstrates the importance of each of the reference models, by comparing their minimal, maximal and average percent point loss against the best performing model out of the ensemble considered per dataset. Some models perform pretty well across all datasets, such as the *DenseNet161* that only performs 1.12% points worse on average than the best performer of the ensemble. However, there exists a dataset where the drop against the best model in that case is still large with 5.73% points.

Despite the fact that there are the above discussed variations of results present among various models, there exist an underling dataset characteristic that saturates the achieved performance of any machine learning-based model. Based on this observation, we can define the theoretical dataset classification difficulty as the saturating accuracy obtained with any best model. However, since at the time of writing we only conducted experiments on a finite list of reference models, we define a stable formulation as mean accuracy obtained over the best $k = 5$ models in the reference experiments of the architectures listed in Table 2. The last column of Table 3 lists the reference dataset classification difficulty number (DCN) that acts as a proxy of the real difficulty that is not measurable.

## 4 Classification difficulty estimation of datasets

In this work, we propose four pipelines to quantify the *difficulty* of image classification datasets. In more detail, we propose dataset scoring functions $r(\mathcal{D})$ to map a dataset $\mathcal{D}$ to a scalar real number, with the goal of scoring different datasets in terms of classification accuracy estimates. For each pipeline, we highlight pros and cons.

### 4.1 Silhouette score

The silhouette score is a well-established metric that compares tightness of same-class samples to separation of different-class samples [43]. Let $i$ be one input sample, $a(i)$ the average Euclidean distance between the sample and all the points $j$ belonging to the same class as $i$, and $b(i)$ the average distance between $i$ and all points $j$ of the closest different class. The *silhouette* of the $i$-th sample is computed as follows [43]:

$$s(i) \quad := \quad \begin{cases} 1 - a(i)/b(i), & \text{if } a(i) < b(i) \\ 0, & \text{if } a(i) = b(i) \\ b(i)/a(i) - 1, & \text{if } a(i) > b(i). \end{cases} \tag{1}$$

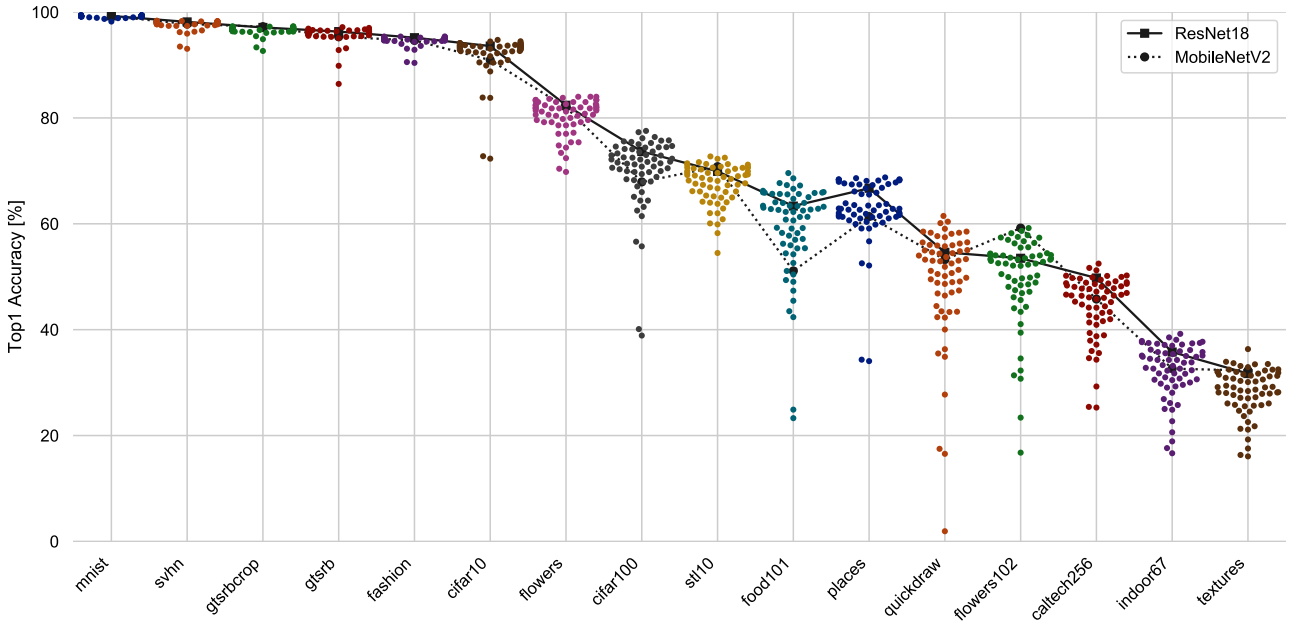


**Fig. 4** Reproduced reference results over an ensemble of established deep learning architectures. Clearly, results cluster at dataset-specific levels, demonstrating the impact of the inherently present classification difficulty imposed by the data of the problem instance. We highlight results obtained with two different architectures, demonstrating that neither one of them is outperforming the others in all cases

**Table 3** Best reference architectures per dataset, and no single architecture is able to outperform the remaining architectures

| Dataset | Best model | Accuracy (%) | Ref DCN (%) |
|---|---|---|---|
| mnist | MobileNetV2 | 99.410 | 99.388 |
| svhn | ResNet152 | 98.371 | 98.291 |
| gtsrbcrop | VGG13 | 97.316 | 97.265 |
| gtsrb | DenseNet121 | 97.150 | 96.903 |
| fashion | PreActResNet101 | 95.390 | 95.324 |
| cifar10 | DenseNet161 | 94.740 | 94.396 |
| flowers | DPN26 | 84.000 | 83.880 |
| cifar100 | DenseNet161 | 77.550 | 76.638 |
| stl10 | GoogLeNet | 72.725 | 72.113 |
| food101 | ResNeXt29_4x64d | 69.584 | 68.082 |
| places | ResNet152 | 68.762 | 68.507 |
| quickdraw | MobileNet | 61.484 | 59.920 |
| flowers102 | VGG11 | 59.197 | 58.289 |
| caltech256 | DenseNet161 | 52.480 | 51.157 |
| indoor67 | ResNeXt29_32x4d | 39.208 | 38.282 |
| textures | GoogLeNet | 36.330 | 34.181 |

The silhouette of one class is defined as the average over all samples belonging to that class, and the overall silhouette score of the full dataset is defined as average over all samples. The definition of the quantities $a(i)$ and $b(i)$ is based on pairwise distances between two samples $i$ and $j$. The silhouette score complexity is $O(\bar{d}n^2)$, where $n$ is the number of samples and $\bar{d}$ is the cost of computing the distance of one pair of samples as mean squared error (MSE) distance in the $\mathbb{R}^{\bar{d}}$. Since the MSE distance in the original domain is a poor measurement for image similarities, we apply first a transformation $\mathbb{R}^d \rightarrow \mathbb{R}^{\bar{d}}$ that maps images into a space that better reflects distances between image pairs.

Table 5 provides details on the applied pipelines. We decided to include a resizing of the images to a small resolution of $8 \times 8$ pixels, applying principal component analysis (PCA) to reduce the dimension to 10, and using a fixed encod-

**Table 4** Comparison of reference models

| Reference Model | Accuracy loss vs. best model | | |
|---|---|---|---|
| | Worst | Mean+/-Std | Best |
| DenseNet161 | 5.73% | 1.12% +/- 1.761% | 0.00% |
| DenseNet169 | 4.01% | 1.34% +/- 1.386% | 0.06% |
| DenseNet121 | 7.49% | 1.68% +/- 1.951% | 0.00% |
| DenseNet201 | 5.59% | 1.70% +/- 1.832% | 0.00% |
| GoogLeNet | 12.38% | 1.72% +/- 3.208% | 0.00% |
| DPN26 | 3.95% | 1.74% +/- 1.343% | 0.00% |
| PreActResNet34 | 6.89% | 2.35% +/- 1.907% | 0.09% |
| PreActResNet18 | 7.35% | 2.37% +/- 2.125% | 0.08% |
| ResNeXt29_32x4d | 10.19% | 2.39% +/- 2.682% | 0.00% |
| ResNet34 | 8.14% | 2.45% +/- 2.578% | 0.01% |
| DPN92 | 8.47% | 2.48% +/- 2.284% | 0.09% |
| ResNet18 | 6.87% | 2.64% +/- 2.399% | 0.14% |
| ResNeXt29_4x64d | 18.02% | 2.66% +/- 4.701% | 0.00% |
| PreActResNet50 | 9.61% | 2.78% +/- 2.614% | 0.07% |
| ResNet101 | 7.39% | 2.83% +/- 2.416% | 0.07% |
| ResNet50 | 11.68% | 2.85% +/- 3.054% | 0.10% |
| VGG13 | 12.60% | 2.91% +/- 3.513% | 0.00% |
| SENet18 | 9.50% | 3.19% +/- 2.934% | 0.10% |
| MobileNetV2 | 7.09% | 3.34% +/- 2.428% | 0.00% |
| PreActResNet152 | 7.47% | 3.35% +/- 2.866% | 0.08% |
| PreActResNet101 | 8.70% | 3.42% +/- 3.218% | 0.00% |
| VGG16 | 11.30% | 3.49% +/- 3.456% | 0.07% |
| ResNet152 | 15.04% | 4.03% +/- 3.987% | 0.00% |
| ResNeXt29_2x64d | 33.74% | 4.11% +/- 8.474% | 0.06% |
| VGG11 | 18.49% | 4.23% +/- 5.081% | 0.00% |
| VGG19 | 14.19% | 4.85% +/- 4.782% | 0.08% |
| MobileNet | 15.16% | 6.45% +/- 5.729% | 0.00% |
| PNASNetB | 24.64% | 8.59% +/- 7.801% | 0.13% |
| PNASNetA | 26.10% | 9.49% +/- 8.073% | 0.30% |
| LeNet | 44.69% | 17.63% +/- 12.738% | 0.56% |

The individual model performance is compared against the best performing model out of the ensemble per dataset and aggregated as worst, mean, and best case over the 16 datasets

**Table 5** Configurations used to compute the silhouette score on datasets

| Score | Transformation | $\bar{d}$ | Distance | Speedup |
|---|---|---|---|---|
| $S_1$ | None | $32^2$ | MSE | 31.3× |
| $S_2$ | None | $32^2$ | DSSIM | 1.0 (Ref) |
| $S_3$ | Resize image | $8^2$ | MSE | 48.4× |
| $S_4$ | Resize image | $8^2$ | DSSIM | 1.3× |
| $S_5$ | PCA | 10 | MSE | 72.8× |
| $S_6$ | Autoencoder | 1000 | MSE | 6.4× |

ing based on a pre-trained CNN inference. We considered as encoder a ResNet-50 [20] network pre-trained on ImageNet [12] to produce generalized per image feature vectors of dimensionality 1000 by taking the output of the last fully connected layer before applying the nonlinearity. Additional to the MSE distance, we used the structural dissimilarity index DSSIM [53] to compare images with a metric that captures spatial information. Due to the squared complexity, we applied heavy subsampling and run all computations with a maximum of 1000 randomly selected samples, resulting in a distance matrix with at most $1M$ entries. Table 5 reports timing among the different pipelines. For fast execution, it is crucial to operate in a low-dimensional space and to use a simple distance metric.

### 4.2 *K*-means clustering

The complexity of the silhouette scores detailed in Sect. 4.1 scales with $n^2$, and computing it is a slow process even after subsampling. In general, the complexity of a deep-learning job is $O(C_{\mathcal{M}} n_{train} E)$, where $E$ is the number of epochs. During one epoch the full training set consisting of $n_{train}$ samples is fed once with a computational cost of $C_{\mathcal{M}}$, which is a model $\mathcal{M}$ dependent constant. Even though complex models might have large computational cost of $C_{\mathcal{M}}$, the asymptotic behavior of a training job is linear in $n$. For this reason, the asymptotic behavior of the silhouette score computation of $n^2$ is outperformed by the actual training job. Competitive scoring metrics should execute faster than a train job itself; thus, we are looking for scores with at most linear complexity in $n$.

We propose to run a (fast) clustering algorithm to produce class labels $\tilde{\mathbf{y}}$ and evaluate the full dataset based on metrics that compare $\tilde{\mathbf{y}}$ against the ground truth labels $\mathbf{y}$. We assess the following known scores: adjusted mutual information [52], adjusted rand index [25], completeness, homogeneity, and the v-measure [42]. Additionally, we propose a tailored score based on the estimation of the confusion matrix built between the cluster indices and the true labels. This score is computed over the permutation of possible labeling configurations of the unsupervised cluster indices that maximizes the trace of the confusion matrix.

### 4.3 Fréchet inception distance based score

The Fréchet inception distance (FID) is widely used as measure to compare the quality of generative adversarial networks (GANs) [21,33] where a comparison of synthetic and real distributions are required to measure the performance of GANs. To that end, the input image samples are feed through an inception network [49] such that each sample is embedded in a learned feature vector space. The resulting embedded vectors of one class are assumed to follow a multivariate Gaussian distribution. The Fréchet distance [14] of the two Gaussian distributions $A$ and $B$ is defined as follows:

$$\|\mu_A - \mu_B\|_2^2 + \mathbb{T}\mathrm{r}(\Sigma_A + \Sigma_B - 2(\Sigma_A \Sigma_B)^{1/2}) \tag{2}$$

where $(\mu_A, \Sigma_A)$ and $(\mu_B, \Sigma_B)$ are the mean vector and covariance matrix of two distributions $A$ and $B$, respectively. Since the FID is defined between two distributions only, a full image classification problem with $C$ classes is characterized by the pairwise FID between classes $i$ and $j$ for $1 < i, j < C$. To summarize and normalize that dataset difficulty as scalar value in an uniform way, we propose the following FID-based score in (2), similar to the definition of the Silhouette score:

$$f := \begin{cases} 1 - F\bar{I}D_{i,i}/FID_{i,j*}, & \text{if } F\bar{I}D_{i,i} < FID_{i,j*} \\ 0, & \text{if } F\bar{I}D_{i,i} = FID_{i,j*} \\ FID_{i,j*}/F\bar{I}D_{i,i} - 1, & \text{if } F\bar{I}D_{i,i} > FID_{i,j*}, \end{cases} \tag{3}$$

whereas $FID_{i,j*}$ denotes the critical distance to the closest neighboring class defined as $FID_{i,j*} := \min_{i \in [1,C] \setminus j} FID_{i,j}$ and $F\bar{I}D_{i,i}$ operates as normalization coefficient. Since the Fréchet distance between two equally distributed Gaussians, $FID_{i,i} \equiv 0$, evaluates to zero, we use $F\bar{I}D_{i,i} = FID_{i',i''}$ where the FID is evaluated over two statistical measurements of $(\mu_{i'}, \Sigma_{i'})$ and $(\mu_{i''}, \Sigma_{i''})$ by computing first- and second-order moments over two disjoint sets of samples belonging to the same class $i$. It turns out to be very beneficial to use the obtained nonzero numerical estimate as normalization coefficient as defined above in order to reduce the obtained Fréchet distances into a normalized score.

The time complexity of the FID based score is $O(n_s C_{Emb} + C^2 C_{FID})$ where the first term is linearly dependent on the number of input samples where one inference of the inception network and the computation of the first- and second-order moments occurs. The second terms has a square dependency on the number of classes since (2) is required to be evaluated pairwise for each class $i$ and $j$. Evaluating (2) costs $C_{FID}$ which itself is determined by the invoked shapes of the matrix, $d$-dimensional mean vector and $d \times d$ sized covariance matrix and mainly determined by the numerical routine computing the blocked Schur algorithm [11] for computing the matrix square root of the product of $\Sigma_A$ and $\Sigma_B$.

**Table 6** Operation count and number of parameters of proposed probe nets

| Probe net | $C = 10$ | | $C = 100$ | |
|---|---|---|---|---|
| | OPs | Weights | OPs | Weights |
| Regular | 0.81M | 11K | 0.86M | 57.5K |
| Narrow | 0.09M | 2K | 0.10M | 13K |
| Wide | 10.34M | 114K | 10.52M | 299K |
| Shallow | 0.24M | 21K | 0.42M | 205K |
| Shallow norm. | 0.06M | 5K | 0.10M | 51K |
| Deep | 1.40M | 100K | 1.41M | 112K |
| Deep norm. | 19.76M | 1576K | 19.81M | 1622K |
| MLPs | 2.90M | 2908K | 3.10M | 3107K |
| Kernel depth | 0.53M | 6K | 4.56M | 384K |
| Length | 1.41M | 118K | 4.39M | 338K |
| ResNet-20 | 40.55M | 271K | 40.56M | 277K |

Even though the matrix square root is optimized with a multi-threaded implementation and runs within tens of seconds for typical invoked problem sizes, the square root dependency to compute all pairwise distances to compute the final score turns that approach quite slow, for example for $C = 100$ classes and $C_{FID} = 10$ sec the total computing time exceeds 24 h.

### 4.4 Probe nets

We propose probe nets that are small and efficient neural networks. We demonstrate that training a probe net and using its accuracy as a dataset difficulty score outperforms the alternative approaches in compute time and difficulty estimation performance. We designed the complexity $C_{\mathcal{M}_{probe}}$ of the architecture of probe nets to be general enough to be applied to any image classification task but considerably faster than training a regular deep learning model $\mathcal{M}$: $C_{\mathcal{M}_{probe}} \ll C_{\mathcal{M}}$. Table 6 reports the operation count and the number of trainable parameters of the proposed probe nets. Additionally, to further speedup the execution time, we demonstrate and discuss in Sect. 6 how stopping the training of probe nets after a few epochs reduces execution time while still achieving good results.

We propose to construct variations of two types of probe nets: *static probe nets* that have a fixed topology and *dynamic probe nets* that scale the topology according to the number of classes. The *regular probe net* consists of three convolutional layers, each followed by batch normalization, max pooling of size $2 \times 2$, and ReLU activations, which are defined element-wise as $x \mapsto \max(0, x)$. We used eight kernels in the first layer and doubled the number of kernels per layer. We provide *wide* and *narrow* variations that scale the number of kernels per layer up and down by $4\times$, respectively. *Shallow* and *deep* variations are obtained by subtracting and adding

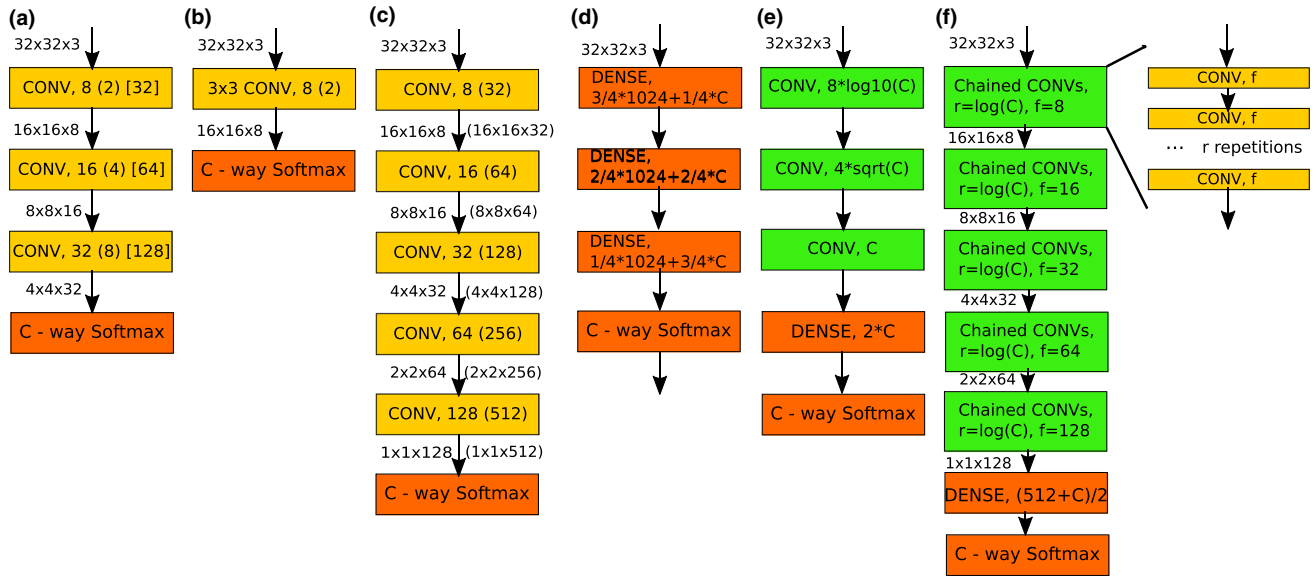


**Fig. 5** *Probe nets*: simple deep learning architectures used to characterize datasets. *Static* networks are shown in **a**–**c**, they only differ in the dense layer connections in the softmax output that ends with a dataset specific number of classes $C$. **a** shows a *regular*, *narrow* and *wide* probe net that differ in kernel depth, **b** shows a *shallow* version and **c** a *deep* version of the net for non and *normalized* kernel depths. *Dynamic* networks, **d**–**f** scale the topology with respect to the number of classes $C$. **d** Consists of dense layers that scale the number of hidden units according to linear weighted sum between input and output dimension, **e** scales kernel depths according to $C$ and **f** scales the number or repetitions of stacked static layers according $C$

two layers, respectively. Since doubling the kernel sizes per layer leads to different tensor shapes between the last convolution and the $C$-way softmax, the *non-normalized shallow* and *deep probe nets* have a considerable different number of trainable parameters. We define *normalized probe networks* to match the number of trainable parameters of the output layer of the *regular probe net*. We construct *dynamic nets* with a more complex topology to account for more classes. This is achieved either by scaling dependent on $C$ the number of hidden units in an multilayer perceptron *mlp*, the number of filters (*filter depth scaled probe nets*), or the number of stacked filters (*length scaled probe net*). Figure 5 shows the ten proposed prob net architectures. We evaluate all ten probe nets as reference but for resource and time efficiency our proposed approach suggest to use the best. The next section presents consistent and superior performance of probe nets over the reference scoring approaches.

## 5 Results

In order to perform a fair evaluation, we fix hyper-parameters throughout the experiments and work with a fixed image resolution of $32 \times 32$ pixels. We evaluate the four proposed dataset difficulty scoring approaches against the reference dataset characterization as given in Table 3. An ideal dataset difficulty score should obey a linear dependency and match the reference DCN. The next subsection states the three alternative scoring pipelines, and Sect. 5.2 shows results achieved with our probe nets. All results are presented as correlations between the proposed score and the reference DCN as listed in Table 3. For each pipeline, we discuss in the following how correlations are affected by tuning configurations of the respective pipeline.

### 5.1 Silhouette, clustering and Fréchet-based scores

Figure 6 shows the obtained results for the silhouette (Sect. 4.1), $k$-means (Sect. 4.2), and Fréchet-based score (Sect. 4.3) correlated against the reference DCN. The three approaches shown are using the versions that produce best results. The results for the silhouette score produce results that range between a correlation from $R^2 = 0.04$ to $R^2 = 0.21$ for the considered configurations defined in Table 5. Best results are obtained by using the structural dissimilarity index DSSIM [53] directly applied in the original domain. Alternatively, using an embedding based on a neural network followed by a mean squared error (MSE) distance metric works equally well, while using PCA to reduce the dimension or directly apply an MSE in the original domain produces weaker correlations.

For the evaluation of the proposed $k$-means-based scoring pipeline (see Sect. 4.2), we cluster the images in $C$ clusters, where $C$ is the known number of categories in

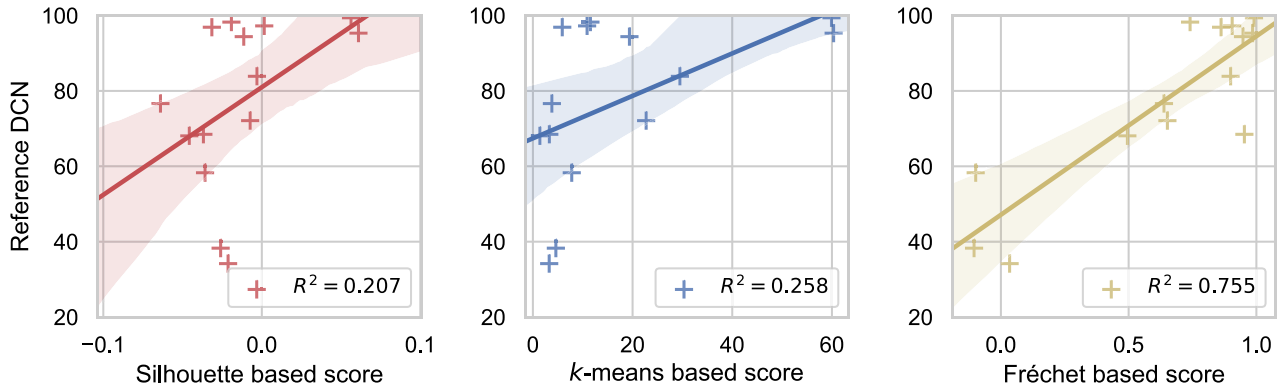


**Fig. 6** Prediction performance of silhouette, clustering and Fréchet-based scores. The $x$-axis reports the obtained score and the $y$-axis reports the reference classification difficulty as in Table 3

the dataset. For a faster convergence, we initialize the centroids with the average image of each class. $k$-means runs based on the Euclidean $L_2$ distance with a stopping tolerance of $10^{-4}$ and a maximum of 300 iterations without random restarts. We tested for preprocessing options: none, resize to smaller image, apply PCA, or use an auto encoder prior to clustering, in junction with the following aggregation metrics *accuracy on the estimated confusion matrix (AECM)*, the *adjusted mutual information* score, the *adjusted random score*, the *v-measure*, the *homogeneity score*, and the *completeness score*. Correlation results range between $R^2 = 0.01$ to $R^2 = 0.26$ where best results are obtained when applying PCA and using AECM. The weak performance of the $k$-means clustering is due to known limitations, such as no global minimum guarantee and a simplistic distance metric that ignores the spatial information. $k$-means clustering-based pipelines are 5.2× (no preprocessing) up to 50.5× (PCA to low dimension) faster than silhouette score-based pipelines (comparison includes the faster MSE timings) when comparing execution times in terms of average compute time per input sample.

The rightmost plot in Fig. 6 shows the Fréchet-based scoring performance. Compared with the two previous approaches, that are weakly correlated with the true dataset difficulty, the FID-based score is strongly correlated with the true difficulty. The evaluated four variants, using an embedding dimensionality of $d = 64$, 192, 768 or 2048, produce correlations that range from $R^2 = 0.71$ to $R^2 = 0.75$. The best Fréchet-based score is achieved with the $d = 192$.

## 5.2 Probe nets

Figure 7 shows all obtained correlations of the ten proposed probe nets against the reference DCN. The probe nets, as presented in Fig. 5, are trained with the same constant configuration and data augmentation parameters as used to produce the reference results. We follow the data augmentation described in [29], and we use the RMSProp [50] optimizer to minimize the average cross-entropy with a learning rate of $10^{-4}$. All evaluations employ the He initialization [19] with a gain factor 1.0 and a constant batch size of 32. Training is run for 100 epochs. The probe nets share a high correlation with the reference DCN that ranges between $R^2 = 0.63$ and $R^2 = 0.95$ and consistently outperform results achieved with the silhouette-based approach or $k$-means-based approach, see Sects. 4.2 and 4.2. Seven out of the ten networks exceed a high correlation of $R^2 > 0.79$ and henceforth outperform the Fréchet-based approach as well, see Sect. 4.3.

Figure 7a shows an increasing correlation of $R^2 = 0.75$ to $R^2 = 0.93$ between *narrow*, *regular*, and *wide* probe nets and the reference DCN. This can be explained by the better generalization ability of the network with more degrees of freedom, at the cost of an increased execution time. Properties of the Probe nets are provided in Table 6. *Deep* probe nets topologies outperform their *shallow* counterparts. This effect is even more prominent in the *normalized* case, Fig. 7b versus d. We observe that a better generalization performance is mainly driven by a larger amount of tunable parameters that comes at the cost of increased execution timings. Figure 7c and e show the results for probe nets that dynamically adapt the architecture topology to the number of classes. The dependency of the architecture on the number of classes implies different execution times on datasets with different number of classes.

## 5.3 Metric alternatives

Figures 6 and 7 present correlations of the best performing configurations per pipeline against the reference DCN as presented in Table 2. Figure 8 justifies that choice and demonstrates the robustness against alternative choices. They include the average accuracy of the ensemble of all reference models as listed in Table 2, the accuracy of specific mod-

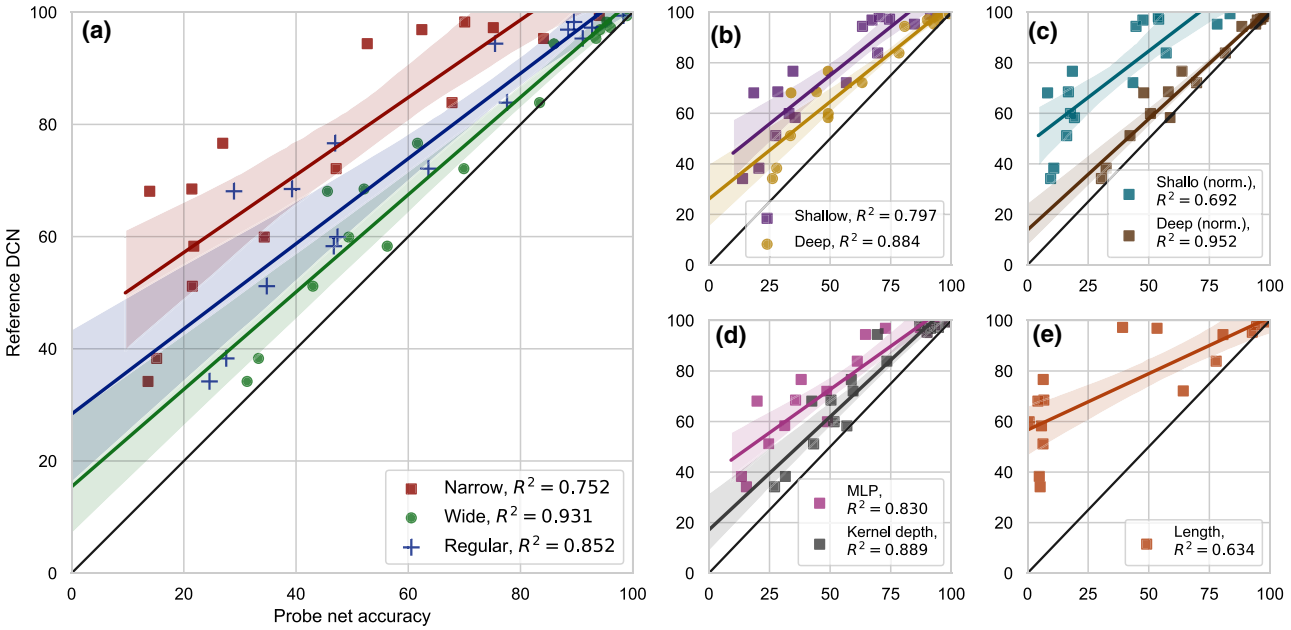


**Fig. 7** Prediction performance of probe nets: the $x$-axis reports the accuracy reached of a converged probe net and the $y$-axis reports the reference DCN. All seven static probe nets, **a** *regular/narrow/wide*, **b** *shallow/deep*, **c** *shallow/deep normalized*, and the three dynamic probe nets, **d** *mlp*, *kernel depth* scaled, and **e** *length* scaled are strongly correlated with the reference DCN

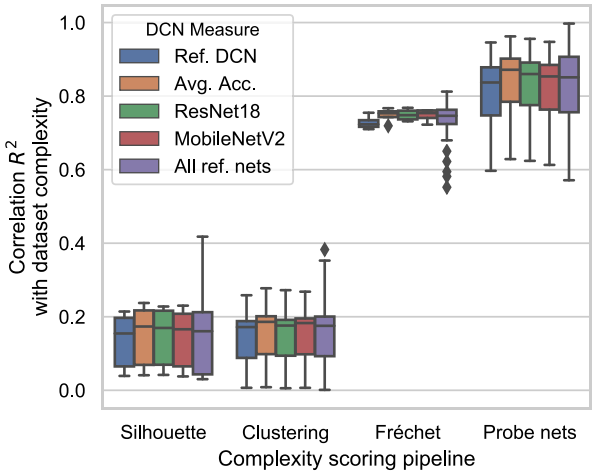


**Fig. 8** The used proxy metric, the reference DCN, is as good as alternative choices that provide consistent results, such as using the average accuracy over all reference models, specific models, or the aggregation of all models. Probe nets outperform the three scoring pipelines

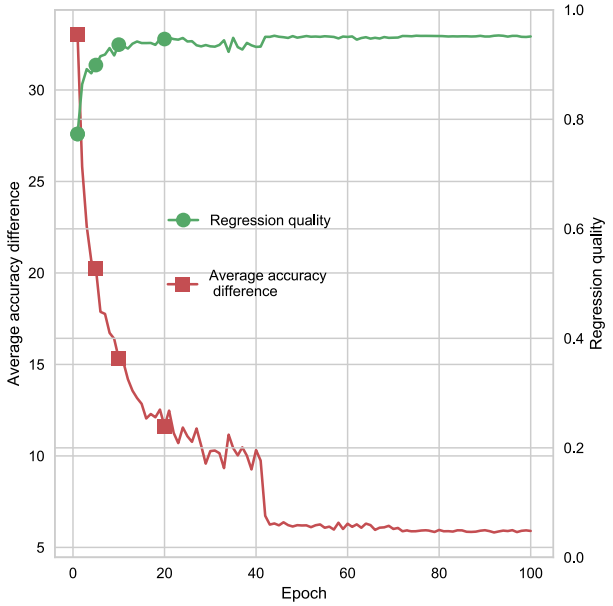


**Fig. 9** Evolution of the prediction quality over training epochs of the *deep normalized* probe net. The regression quality reaches high values within a few epochs, while the average accuracy difference between the probe net and the reference DCN is further decreased for longer training

els, and the aggregated distribution of correlations computed against all reference models. As Fig. 8 shows all options yield consistent correlations validating our proposed reference DCN as a suitable proxy for the reference complexity of the classification task. Probe nets are consistently providing higher correlations than the tree alternative scores. Those findings are robust against different configurations in the pipelines and the proxy metric of the dataset complexity.

## 6 Efficient evaluation of probe nets

As presented in Sect. 5.2, probe nets have a good predictive behavior of what a reference network achieves on a

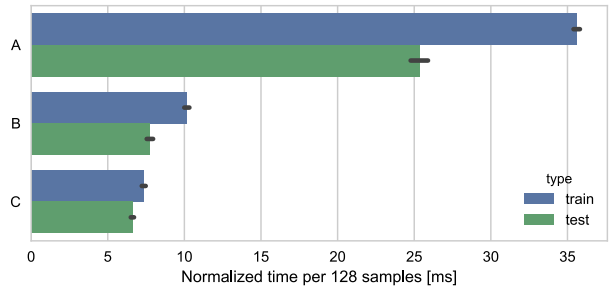


**Fig. 10** Normalized execution time of a *deep normalized* probe net. Scenario A uses standard settings, while in scenario B the average execution time is significantly reduced if the probe net is trained with a large batch size of 1024 samples and 8 threads are used to perform the on-the-fly standard preprocessing, including padding and random cropping, random horizontal flips and normalization. In scenario C) processing times are further reduced if the preprocessing is minimized to the required cast and transformation from the CPU to the GPU

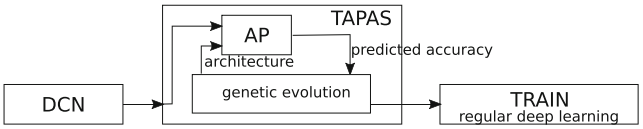


**Fig. 11** System overview of an efficient architecture search. First, the dataset is characterized, second TAPAS uses a genetic algorithm that predicts performance of candidate architectures to find a suited architecture, and third, the selected architecture is finally trained on the given dataset

given dataset. However, that information is more valuable if it can be computed faster than training large models. The way probe nets are constructed give them an inherent computational benefit over the full model. In addition, we exploit early stopping of the learning to further reduce the computational time of the probe net. Note that we can stop the probe net before convergence, since we are interested in the learning trend that characterizes the problem's difficulty, not in the final accuracy. Figure 9 shows how the prediction quality improves for a *deep normalized* probe net with an increasing amount of epochs for which it is trained on all datasets. Even within the first epoch, the regression quality outperforms the FID-based approach. The mean accuracy difference between the probe nets and the reference DCN (trained till convergence) is further decreased, meaning that the probe nets are not yet converged and are still increasing their own classification performance.

We evaluate all probe nets with the same settings, using a batch size of 128 samples and 2 threads during on-the-fly preprocessing, as the reference network timings shown in Fig. 9. In the case of the reference, the average batch time was 194 ms and was compute bound by the operations performed on the GPU for most of the models. However, we observed that for small models, and this is especially relevant for the designed probe nets, the GPU might not be fully nor optimally utilized since relative overheads of batch preparation and loading have a relative higher impact against the lowered GPU workload of very small networks. In such settings, the overall timing performance is strongly dependent on implementation details and the underling hardware. Since that is not the case for medium and large sized networks, we used a standardized setting of a multi-threaded data parallel loader that performs batch preparation on the CPU side. All measurements include batch preparation times consisting of padding, random cropping, random horizontal flipping and normalization. Since CPU and GPU operation are performed in parallel, most of the batch preparation times are hidden behind the GPU computation or are small compared to the GPU workload such that they do not matter. However, for small networks, such as the probe nets, we found it beneficial to fine tune the settings. Running with larger batch sizes helps to reduce kernel overheads and leads to better utilization and execution performance. We tested batch sizes of powers of two $2^i$ from $2^0 = 1$ up to $2^{13} = 8192$ and observed best performance for a batch size of 1024. In the performance measurement we assumed a dataset size of 8192 samples, e.g., the number of batches measured is given as $2^{13-i}$, which is relevant for filling/ flushing effects of the pipeline. For example, the corner case of the batch size 8192 causes only one batch triggered to be executed, which cause the GPU to stall and wait till the first batch preparation has fully finished. Figure 10 shows the normalized execution time for 128 samples with A) standard settings, B) optimized settings using a larger batch size of 1024 samples and using 8 instead of 2 threads and C) the optimized settings but with a minimal batch preparation pipeline that only casts CPU float arrays into GPU allocated tensors. The optimized settings B) allow to run the *deep normalized* probe net within 10.2 ms per 128 samples, that is 19.4× faster than the average execution speed of the reference models. The training speed of the *CIFAR10* dataset results of about four seconds per epoch and completes the 100 epochs within seven minutes.

## 7 Application scenario: dataset characterization for fast architecture search

In this section, we demonstrate how the dataset characterization enables efficient architecture search. An in-depth discussion of the broad field of automated architecture search is widely covered in literature [1,5,34,38,41,55,57,58]. Automated architecture search enables to discover new network models that might outperform reference models on a given dataset and henceforth address the discovered problem that arises when just applying given reference networks as done in Sect. 3. However, in a traditional approach a large amount of

**Table 7** Run time complexity of the main stages of the TAPAS-based pipeline as depicted in Fig. 11

| Stage | Complexity | Typical time (*CIFAR10*) |
|---|---|---|
| DCN | $O(n_{train} C_{DCN} E_{DCN})$ | $< 10$ min |
| TAPAS | $O(n_{Candidates} C_{TAP})$ | $< 5$ min |
| TRAIN | $O(n_{train} C_{TRAIN} E_{TRAIN})$ | $\sim 2$h |

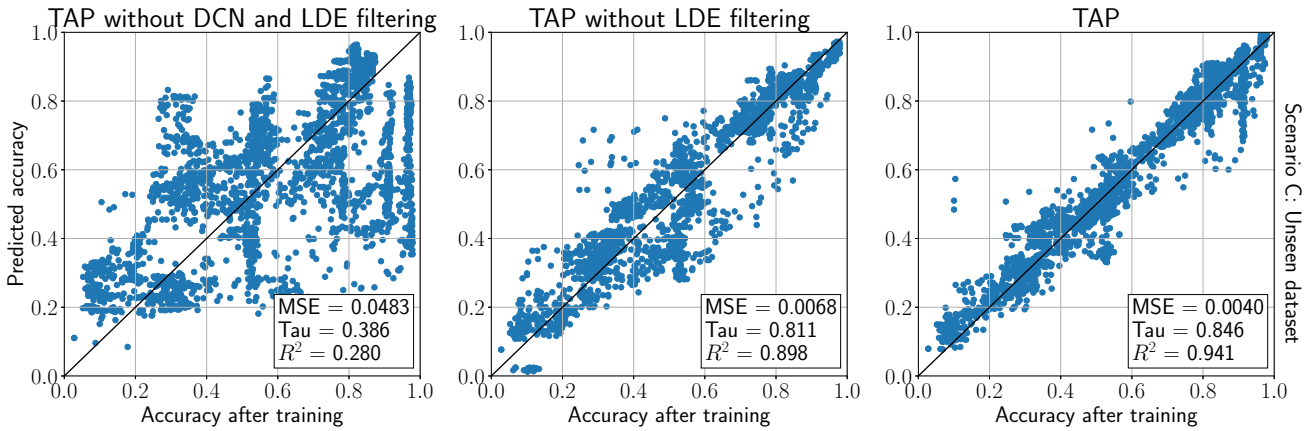


**Fig. 12** Reprinted with permission [26]. Predicted vs real performance (i.e., after training) of random architectures. Left plot: TAP trained without DCN or LDE pre-filtering. Middle plot: TAP trained with DCN, but LDE is not pre-filtered. Right plot: TAP trained only on LDE experiments with similar dataset difficulty

candidate networks has to be fully trained on a given dataset, either requiring a vast amount of computing resource or taking a long time to complete. To that end, approaches that accelerate learning during architecture search have been proposed. Early stopping based on learning curve predictors, or transferred learning from model weights to the next candidate model was applied with success. Probe nets enable fast execution and good classification difficulty estimation that provide insights into datasets. Classification difficulty estimation enables to select a problem tailored model prior to invest resources and time to fully train all alternatives. Using the knowledge of dataset difficulty estimation is a key contribution to bias and optimize architecture searches toward models of fitted complexity. We recently proposed a method called train-less accuracy predictor for architecture search (TAPAS) that demonstrates how the probe nets contribute to bias and perform an architecture search. In this section, we present TAPAS that is able to perform the complete architecture search without training or retraining during the architecture search [26] at all by tuning its search based on the DCN.

## 7.1 TAPAS system setup

Figure 11 shows the high level system that is invoked for the train-less architecture search of TAPAS. In this section, we provide a high-level system description focusing on the use of the dataset characterization, whereas a detail technical discussion of the internal operation of TAPAS is found in the literature [26]. The core of TAPAS performs a genetic evolution algorithm, similar as proposed [41], however with the key differentiation that none of the candidate models are trained. In contrast, an accuracy predictor (AP) has been trained beforehand to predict the accuracy given a network topology description and the dataset characterization. A prediction-based driven genetic algorithm can be executed several orders of magnitude faster than the same algorithm that would require to train each candidate network. In order to work properly, the accuracy predictor requires to gain and reuse knowledge about the dataset in order to accurately predict model performance for different datasets. To that end, the dataset characterization is used twice, offline to train the AP, and online as input to the AP to bias all predictions to that data at hand. Given a new dataset instance, as shown in Fig. 11, the dataset characterization is computed as first step, since it is required as input for TAPAS to work. After TAPAS search yields a valid network, it is trained on the given data and returned to the user. Table 7 summarizes the time complexity for each of the three steps. TAPAS execution time does not depend on the size of the input dataset nor of the run time complexity of the models it predicts performance for. It solely depends on the number of evaluations, typically a constant that is given due to the population size and the number of iterations in the evolution algorithm, and the complexity of the encoded candidate model description. In all our experiments, TAPAS was able to execute in less than a few

**Table 8** Obtained accuracy with TAPAS-based pipeline as depicted in Fig. 11

| Dataset | Variation (%) | Best (%) | Best+ (%) |
|---|---|---|---|
| mnist | 99.19 ± 0.08 | 99.31 | 99.31 |
| svhn | 97.62 ± 0.19 | 98.12 | 98.12 |
| gtsrbcrop | 96.19 ± 0.39 | 96.57 | 96.57 |
| gtsrb | 95.71 ± 0.56 | 96.37 | 96.37 |
| fashion | 93.94 ± 0.60 | 94.58 | 94.58 |
| cifar10 | 90.57 ± 1.06 | 91.80 | 91.80 |
| flowers | 74.68 ± 2.82 | 79.80 | **81.60** |
| cifar100 | 59.46 ± 5.18 | 65.30 | 65.30 |
| stl10 | 61.96 ± 2.60 | 65.44 | **69.72** |
| food101 | 46.60 ± 8.91 | 57.90 | 57.90 |
| places | 57.76 ± 4.85 | 65.08 | 65.08 |
| quickdraw | 55.06 ± 4.42 | 65.41 | 65.41 |
| flowers102 | 26.74 ± 10.14 | 39.49 | **58.72** |
| caltech256 | 32.57 ± 3.91 | 38.40 | **42.26** |
| indoor67 | 21.02 ± 2.45 | 24.50 | **32.41** |
| textures | 13.05 ± 4.24 | 20.69 | **30.48** |

Bold values indicate zero-cost improvements by using the probe net rather than the generated architecture

minutes. Both the DCN computation and the training time of the candidate model are proportional to the dataset size, the invoked model complexity, and the number of epochs the models are trained for. Since both contributions are of the same order and contribute to most of the time spent in the overall pipeline, the DCN computation becomes time critical. For the DCN stage, the probe net defines the model complexity and is designed to be lightweight and constant. In contrast, the complexity of the candidate model is itself a function of the resulting architecture found by TAPAS causing model-based execution time variations among different runs. As discussed in Sect. 6, the DCN can be computed with a low number of epochs resulting in superior execution time advantages over typical training settings used to train the candidate model.

## 7.2 Results

Figure 12 depicts the dependency of the dataset characterization on the TAP performance [26] when applied to completely unseen datasets. To that end, TAPs performance was measured on one unseen datasets while the TAP has been trained on the remaining datasets. The full experiment is cross-validated among all datasets. Without the DCN, the TAP is not able to well predict the accuracy of candidate models on the given dataset, resulting in a very loose cross-correlation of $r^2 = 0.28$, see left plot of Fig. 12. However, if the DCN is feed and used by the TAP, its performance is increased to a high cross-correlation. We observed that it turns out to be beneficial to use the DCN once more in order to pre-filter the available experiments in order to restrict the ground truth used to train TAP to experiments that are relevant, by selecting entries that are performed on datasets with similar dataset characterization values. The later option, relying twice on the DCN, results in high correlation between the predicted and the actual accuracy reached over the networks tested of about $R^2 = 0.94$.

For completeness, Table 8 shows the accuracy obtained with the TAPAS framework. For each dataset, we repeated the pipeline $n = 10$ times generating $n$ different architectures per dataset. Average and standard deviation over the $n$ repetitions are given in the first column. The second column reports the best accuracy reached, and the last column reports the best results reached when reusing the probe net rather than the generated architecture in cases where the probe net outperforms the solution found by the architecture search. Reusing the probe net is a zero-cost improvement as this does not add additional computing time since the probe net is already trained within the regular high level execution flow as described in Fig. 11. This optimization turns out to be helpful on very small datasets where TAPAS might find too complex network architectures that then over-fit to the dataset. The last column of Table 8 reports the result of this optimization.

An extended study of architecture search and improvements thereon is out of scope for that work. The results presented in Table 8 are all based on TAPAS assumptions and implementation details [26]. TAPAS internally uses a block-based encoding to map network models to inputs processed by the AP. The setup of this encoding enforces block-sequential structure of all considered networks. Even though TAPAS considers already a large search space, some of the reference models considered in Sect. 3 are outside the reachable search space of TAPAS. That present limitation renders a direct comparison of models from Sect. 3 with current results unfair and is henceforth omitted.

In this section, we demonstrated how the dataset characterization fine-tunes the core part of an accuracy predictor. Without any knowledge about the data, it would not be possible to generate data and architecture specific predictions. The TAPAS pipeline as depicted in Fig. 11 is able to transfer knowledge about data and architecture to the use case-specific situation even when applied with completely unseen datasets. In contrast to traditional architecture searches that rely on fully or partially training produced candidates, TAPAS mechanism is based on prediction-based construction of suited architectures enabled through the dataset characterization. The training-free design enables to run the TAPAS-based pipeline as shown in Fig. 11 multiple order of magnitudes faster than traditional architecture search approaches.

## 8 Conclusion

We formulated the question to compute a score among datasets that reflect their inherent classification difficulty. We suggested four processing pipelines, a silhouette-based score, a $k$-means clustering-based, a Fréchet inception distance-based score and our probe net-based evaluation pipeline. The main drawback of the silhouette-based approach is the high complexity, which scales with the squared number of samples. We proposed efficient score computing pipelines based on $k$-means, Fréchet inception distance (FID) and probe nets that scale linear in the number of samples. $k$-means delivers results one complexity class faster and with slightly better prediction quality as the silhouette approach, reaching a weak correlation with reference models of $R^2 = 0.26$. The FID-based approach reaches a high correlation of $R^2 = 0.76$ but at the drawback that its compute times is determined by solving a matrix square root problem $C^2$ times.

Finally, we developed the probe nets, which are small networks, and apply standard deep learning techniques in order to compute predictions that are strongly correlated with the reference DCN reaching correlations of $R^2 = 0.95$. Even the worst performing probe net outperforms silhouette and $k$-means-based scoring with a wide quality margin. We further evaluated the fact of early stopping to reduce the data score evaluation time and observed little to no performance drop. Leveraging the small architectures of probe nets and early stopping allows to perform dataset scoring 97× faster than the required training time of the average reference model.

**Acknowledgements** Open access funding provided by Swiss Federal Institute of Technology Zurich. We would like to thank Dr. Dario Garcia Gasulla from the Barcelona Supercomputing Center for discussion and advise.



## References

1. Baker, B., Gupta, O., Naik, N., Raskar, R.: Designing neural network architectures using reinforcement learning. CoRR (2016). arXiv:1611.02167
2. Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., Vaughan, J.W.: A theory of learning from different domains. Mach. Learn. **79**(1–2), 151–175 (2010)
3. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. J. Mach. Learn. Res. **13**, 281–305 (2012)
4. Bossard, L., Guillaumin, M., Van Gool, L.: Food-101 - mining discriminative components with random forests. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) Computer Vision - ECCV 2014, pp. 446–461. Springer International Publishing, Cham (2014)
5. Cai, H., Chen, T., Zhang, W., Yu, Y., Wang, J.: Efficient architecture search by network transformation. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
6. Chen, L., Wang, R., Yang, J., Xue, L., Hu, M.: Multi-label image classification with recurrently learning semantic dependencies. Vis. Comput. **35**(10), 1361–1371 (2019)
7. Chen, W., Huang, H., Peng, S., Zhou, C., Zhang, C.: Yolo-face: a real-time face detector. Vis. Comput. pp. 1–9 (2020)
8. Cimpoi, M., Maji, S., Kokkinos, I., Mohamed, S., Vedaldi, A.: Describing textures in the wild. In: Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR '14, pp. 3606–3613. IEEE Computer Society, Washington, DC, USA (2014). 10.1109/CVPR.2014.461
9. Coates, A., Ng, A., Lee, H.: An analysis of single-layer networks in unsupervised feature learning. In: Gordon, G., Dunson, D., Dudík, M. (eds.) Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 15, pp. 215–223. PMLR, Fort Lauderdale, FL, USA (2011). http://proceedings.mlr.press/v15/coates11a.html
10. Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R., Bengio, Y.: Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. arXiv preprint arXiv:1602.02830 (2016)
11. Deadman, E., Higham, N.J., Ralha, R.: Blocked schur algorithms for computing the matrix square root. In: International Workshop on Applied Parallel Computing, pp. 171–182. Springer, Berlin (2012)
12. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: a large-scale hierarchical image database. In: IEEE CVPR, pp. 248–255 (2009)
13. Deng, L.: The mnist database of handwritten digit images for machine learning research [best of the web]. IEEE Signal Process. Mag. **29**(6), 141–142 (2012)
14. Dowson, D., Landau, B.: The fréchet distance between multivariate normal distributions. J. Multivar. Anal. **12**(3), 450–455 (1982)
15. Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., Lempitsky, V.: Domain-adversarial training of neural networks. J. Mach. Learn. Res. **17**(1), 2030–2096 (2016)
16. Griffin, G., Holub, A., Perona, P.: Caltech-256 object category dataset (2007)
17. Gupta, S., Agrawal, A., Gopalakrishnan, K., Narayanan, P.: Deep learning with limited numerical precision. In: International Conference on Machine Learning, pp. 1737–1746 (2015)
18. Hanzhang H., Debadeepta Dey, M.H.J.A.B.: Anytime neural network: a versatile trade-off between computation and accuracy (2017)
19. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE international conference on computer vision, pp. 1026–1034 (2015)
20. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
21. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. In: Advances in Neural Information Processing Systems, pp. 6626–6637 (2017)

22. Ho, T.K., Basu, M.: Complexity measures of supervised classification problems. IEEE Trans. Pattern Anal. Mach. Intell. **3**, 289–300 (2002)
23. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. CoRR (2017). arXiv:1704.04861
24. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4700–4708 (2017)
25. Hubert, L., Arabie, P.: Comparing partitions. J. Classif **2**(1), 193–218 (1985)
26. Istrate, R., Scheidegger, F., Mariani, G., Nikolopoulos, D., Bekas, C., Malossi, A.C.I.: Tapas: Train-less accuracy predictor for architecture search (2018)
27. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images (2009)
28. Kumar, A., Sattigeri, P., Wadhawan, K., Karlinsky, L., Feris, R., Freeman, B., Wornell, G.: Co-regularized alignment for unsupervised domain adaptation. In: Advances in Neural Information Processing Systems, pp. 9367–9378 (2018)
29. Lee, C.Y., Xie, S., Gallagher, P., Zhang, Z., Tu, Z.: Deeply-supervised nets. In: Artificial Intelligence and Statistics, pp. 562–570 (2015)
30. Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., Talwalkar, A.: Hyperband: Bandit-based configuration evaluation for hyperparameter optimization (2016)
31. Liu, C., Zoph, B., Neumann, M., Shlens, J., Hua, W., Li, L.J., Fei-Fei, L., Yuille, A., Huang, J., Murphy, K.: Progressive neural architecture search. In: The European Conference on Computer Vision (ECCV) (2018)
32. Luciano, L., Hamza, A.B.: Deep similarity network fusion for 3d shape classification. Vis. Comput **35**(6–8), 1171–1180 (2019)
33. Lucic, M., Kurach, K., Michalski, M., Gelly, S., Bousquet, O.: Are gans created equal? A large-scale study. In: Advances in neural information processing systems, pp. 698–707 (2018)
34. Miikkulainen, R., Liang, J., Meyerson, E., Rawal, A., Fink, D., Francon, O., Raju, B., Shahrzad, H., Navruzyan, A., Duffy, N., et al.: Evolving deep neural networks. In: Artificial Intelligence in the Age of Neural Networks and Brain Computing, pp. 293–312. Elsevier (2019)
35. Mundhenk, T.N., Konjevod, G., Sakla, W.A., Boakye, K.: A large contextual dataset for classification, detection and counting of cars with deep learning. In: European Conference on Computer Vision, pp. 785–800. Springer (2016)
36. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning. In: NIPS Workshop on Deep Learning and Unsupervised Feature Learning, vol. 2011, p. 5 (2011)
37. Nilsback, M.E., Zisserman, A.: Automated flower classification over a large number of classes. In: 2008 Sixth Indian Conference on Computer Vision, Graphics Image Processing, pp. 722–729 (2008)
38. Pham, H., Guan, M.Y., Zoph, B., Le, Q.V., Dean, J.: Efficient neural architecture search via parameter sharing. CoRR (2018). arXiv:1802.03268
39. Pourashraf, P., Tomuro, N.: Use of a large image repository to enhance domain dataset for flyer classification. In: International Symposium on Visual Computing, pp. 609–617. Springer, Berlin (2015)
40. Quattoni, A., Torralba, A.: Recognizing indoor scenes. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 413–420 (2009). 10.1109/CVPR.2009.5206537
41. Real, E., Moore, S., Selle, A., Saxena, S., Suematsu, Y.L., Tan, J., Le, Q., Kurakin, A.: Large-scale evolution of image classifiers (2017)
42. Rosenberg, A., Hirschberg, J.: V-measure: A conditional entropy-based external cluster evaluation measure. In: Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL) (2007)
43. Rousseeuw, P.J.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. J. Comput. Appl. Math. **20**, 53–65 (1987)
44. Scheidegger, F., Istrate, R., Mariani, G., Benini, L., Bekas, C., Malossi, C.: Efficient image dataset classification difficulty estimation for predicting deep-learning accuracy (2018)
45. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
46. Snoek, J., Larochelle, H., Adams, R.P.: Practical bayesian optimization of machine learning algorithms. In: Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 25, pp. 2951–2959. Curran Associates, Inc. (2012). http://papers.nips.cc/paper/4522-practical-bayesian-optimization-of-machine-learning-algorithms.pdf
47. Stallkamp, J., Schlipsing, M., Salmen, J., Igel, C.: The german traffic sign recognition benchmark: A multi-class classification competition. In: The 2011 International Joint Conference on Neural Networks, pp. 1453–1460 (2011). 10.1109/IJCNN.2011.6033395
48. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–9 (2015)
49. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
50. Tieleman, T., Hinton, G.: Lecture 6.5-rmsprop, coursera: Neural networks for machine learning. University of Toronto, Technical Report (2012)
51. Tudor Ionescu, R., Alexe, B., Leordeanu, M., Popescu, M., Papadopoulos, D.P., Ferrari, V.: How hard can it be? estimating the difficulty of visual search in an image. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2157–2166 (2016)
52. Vinh, N.X., Epps, J., Bailey, J.: Information theoretic measures for clusterings comparison: variants, properties, normalization and correction for chance. J. Mach. Learn. Res. **11**, 2837–2854 (2010)
53. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE Trans. Image Process. **13**(4), 600–612 (2004). https://doi.org/10.1109/TIP.2003.819861
54. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms (2017)
55. Xie, L., Yuille, A.: Genetic cnn. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1379–1388 (2017)
56. Zhou, B., Lapedriza, A., Khosla, A., Oliva, A., Torralba, A.: Places: A 10 million image database for scene recognition. IEEE Trans. Pattern Anal. Mach. Intell. (2017)
57. Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning. CoRR (2016). arXiv:1611.01578
58. Zoph, B., Vasudevan, V., Shlens, J., Le, Q.V.: Learning transferable architectures for scalable image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 8697–8710 (2018)

**Florian Scheidegger** achieved a Master of Science ETH in Electrical Engineering and Information Technology in 2017 with a main focus on software and hardware development for high-perfor mance data processing applications. He worked for five months at the Integrated Systems Laboratory of ETH developing a spatiotemporal video pipeline in first part, followed by a work that uses neuronal nets to classify videos. In January 2017, he started as PhD at IBM research Zürich for the OPRECOMP project.

**Roxana Istrate** is currently a research staff member at IBM Re search in Zurich focused on AI. She graduated from the Polytechnic University of Bucharest, Faculty of Computer Science, in 2015 and joined IBM Research the same year as a Great Minds intern. She contributed to winning the 2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS) best paper award. After completing her internship, Roxana started her PhD with IBM Research in collaboration with the Queens University of Belfast. Her focus lies on automating the architectural design of convolutional neural networks for image classification.

**Giovanni Mariani** is a Senior Deep Learning Engineer at Qualcomm since the beginning of 2019. Before, he worked as postdoctoral resea rcher in the Foundations of Cognitive Solutions group of IBM. Prior to that, he was a postdoctoral research at the ASTRON & IBM Center for Exascale Technology located in Dwingeloo, the Netherlands. He received his Ph.D. degree on 2011 from Universitá della Svizzera Italiana for his work on design-time and run-time optimization of multi-core systems.

**Luca Benini** is the chair of digital Circuits and Systems at D-ITET ETHZ. He received a Ph.D. degree in electrical engineering from Stanford University in 1997. He has served as Chief Architect for the Platform2012/STHORM project in STmicroelectronics, Grenoble, in the period 2009-2013. He is also a professor at University of Bologna and has held visiting and consulting resea rcher positions at EPFL, IMEC, Hewlett Packard Laboratories, Stanford University. Dr. Benini's research interests are in the design of energy-efficient digital systems with special emphasis on ultra-low-power system-on-chip and green HPC systems. He is also active in the area of smart sensors and sensor networks for consumer, biomedical, and Internet-of-Things applications. In these areas, he has coordinated tens of funded projects, including an ERC Advanced Grant on Multi-scale thermal management of Computing Systems. He has been general chair of the Design Automation and Test in Europe Conference, of the Network on Chip Symposium and of the International Symposium on Low Power Electronics and Design. He is Associate Editor of the IEEE Transactions on Computer-Aided Design of Circuits and Systems and the ACM Transactions on Embedded Computing Systems. He has published more than 700 papers in peerreviewed international journals and conferences, four books, and several book chapters. He is a Fellow of the IEEE and the ACM and a member of the Academia Europaea. He is the recipient of the 2016 IEEE CAS Mac Van Valkenburg award.

**Costas Bekas** is managing the Foundations of Cognitive Computing group at IBM Research-Zurich. He received B. Eng., Msc and PhD diplomas, all from the Computer Engineering & Informatics Department, University of Patras, Greece, in 1998, 2001 and 2003, respectively, working under the supervsion of E. Gallopoulos. Between 2003 and 2005, he worked as a postdoctoral associate with Prof. Yousef Saad at the Computer Science & Engineering Department, University of Minnesota, USA. He has been with IBM since September 2005. Costas's main research interests span Scalable systems for AI and Knowledge Ingestion, HPC and new computing paradigms. Costas is a recipient of the PRACE 2012 award and the ACM Gordon Bell 2013 and 2015 prizes.

**Cristiano Malossi** received his B.Sc. in Aerospace Engineering and his M.Sc. in Aeronautical Engineering from the Politecnico di Milano (Italy) in 2004 and 2007, respectively. After a year of focus on computational geology problems in collaboration with ENI, he moved to Switzerland where in 2012 he got his Ph.D. in Applied Mathematics from the Swiss Federal Institute of Technology in Lausanne (EPFL), with a thesis focused on the development of algorithms and mathematical methods for the numerical simulation of cardiovascular problems. After winning the IBM Research Prize for his PhD thesis, in July 2013 Cristiano joined IBM Research-Zurich in the Foundations of Cognitive Solutions group. Cristiano is a recipient of the 2015 ACM Gordon Bell Prize and 2016 IPDPS Best Paper Award. Since 2017 he is coordinator of the FET-H2020 Open transPREcision COMPuting (OPRECOMP) project. Since 2018 he is Research lead of NeuNetS. His main research interests include: AI, AI automation, deep learning & machine learning, high-performance computing, transprecision & energy-aware computing, numerical analysis, computational fluid dynamics, aircraft design, cardiovascular simulations, and computational geology.