Meta-learning Convolutional Neural Architectures for Multi-target Concrete Defect Classification with the COncrete DEfect BRidge IMage Dataset

Martin Mundt[1*], Sagnik Majumder[1], Sreenivas Murali[1*], Panagiotis Panetsos[2], Visvanathan Ramesh[1*]
1. Goethe University 2. Egnatia Odos A. E.
{mmundt, vramesh}@em.uni-frankfurt.de {majumder, murali}@ccc.cs.uni-frankfurt.de
ppane@egnatia.gr

Abstract

Recognition of defects in concrete infrastructure, especially in bridges, is a costly and time consuming crucial first step in the assessment of the structural integrity. Large variation in appearance of the concrete material, changing illumination and weather conditions, a variety of possible surface markings as well as the possibility for different types of defects to overlap, make it a challenging real-world task. In this work we introduce the novel COncrete DEfect BRidge IMage dataset (CODEBRIM) for multi-target classification of five commonly appearing concrete defects. We investigate and compare two reinforcement learning based meta-learning approaches, MetaQNN and efficient neural architecture search, to find suitable convolutional neural network architectures for this challenging multi-class multi-target task. We show that learned architectures have fewer overall parameters in addition to yielding better multi-target accuracy in comparison to popular neural architectures from the literature evaluated in the context of our application.

1. Introduction

To assess a concrete bridge's structural safety, it is desirable to determine the level of degradation by accurately recognizing all defect types. Defects tend to be small with respect to bridge elements and often occur simultaneously with overlap of defect categories. Although one could imagine treating each defect category independently, overlapping defects are more severe with respect to the structural safety. The requirement to recognize these multi-class multi-target defects forms the basis for a challenging real-world task that is further complicated by a variety of environmental factors. Concrete, as a composite material, has a wide range of variation in surface reflectance, roughness, color and, in some cases, applied surface coatings. Changing lighting conditions, weather dependent wetness of the surface and a diverse set of safety irrelevant surface alterations like small holes, markings, stains or graffiti, add to the factors of variation. This necessitates computer vision techniques that are capable of addressing such rich appearance spaces.

Deep learning techniques in conjunction with labelled datasets have turned out to be ideal candidates for recognition tasks of similar complexity. Especially convolutional neural networks (CNNs) [21, 32, 1, 37, 16] have been shown to excel at object and material recognition benchmarks [29, 10, 35, 3]. Unfortunately, defect recognition in concrete bridges is largely yet to benefit from deep learning approaches. Due to the necessity of expert knowledge in the annotation process along with tedious image acquisition, the task is traditionally focused on cracks with algorithms based on domain specific modelling or manual inspection by a human. Recently, datasets [31, 36, 26] and corresponding deep learning applications [36, 23, 18, 8] have presented significant efforts towards data-driven approaches in this domain. Their work focuses on cracks as only a subset of structurally relevant defects and concentrates on CNNs proposed in the object recognition literature, that might not be the best choice for material defect recognition.

In this work we address two crucial open aspects of concrete defect recognition: the establishment of a labelled multi-target dataset with overlapping defect categories for use in machine learning and the design of deep neural networks that are tailored to the task. For this purpose we introduce our novel COncrete DEfect BRidge IMage (CODEBRIM) dataset and employ meta-learning of CNN architectures specific to multi-class multi-target defect classification. CODEBRIM features six mutually non-exclusive classes: crack, spallation, efflorescence, exposed bars, corrosion (stains) and non-defective background. Our images were acquired at high-resolution, partially using an unmanned aerial vehicle (UAV) to gain close-range access, and feature varying scale and context. We evaluate a variety of best-practice CNN architectures [21, 32, 1, 37, 16] in the literature on the CODEBRIM's multi-target defect

* work conducted while at Frankfurt Institute for Advanced Studies

recognition task. We show that meta-learned neural architectures achieve equivalent or better accuracies, while being more parameter efficient, by investigating and comparing two reinforcement learning neural architecture search approaches: MetaQNN [2] and "efficient neural architecture search" (ENAS) [27]. The CODEBRIM dataset is publicly available at: https://doi.org/10.5281/zenodo.2620293 . We also make the code for training the CNN baselines and both meta-learning techniques available open-source at: https://github.com/MrtnMndt/meta-learning-CODEBRIM . To summarize our contributions:

- We introduce a novel high-resolution multi-class multi-target dataset featuring images of defects in context of concrete bridges.
- We evaluate and compare best-practice CNN architectures for the task of multi-target defect classification.
- We adapt and contrast two reinforcement learning based architecture search methods, MetaQNN and ENAS, on our multi-target scenario. We show how resulting meta-learned architectures from both methods improve the presented task in terms of higher accuracy and lower model parameter count.

2. Prior and related work

Datasets. Image classification and object detection benchmarks predominantly focus on the single-target scenario. Popular examples are the ImageNet [29], Pascal VOC [10] or the scene understanding SUN dataset [35], where the task is to assign a specific class to an image, area or pixel. Much of the recent computer vision deep learning research is built upon improvements based on these publicly available datasets. The "materials in context" database (MINC) [3] followed in spirit and has created a dataset for material and texture related recognition tasks. To a large degree MINC has extended previous datasets and applications built upon prior work of the (CUReT) database [9], the FMD dataset [30] and KTH-TIPS [11, 5]. With respect to defects in concrete structures, or bridges in particular, openly available datasets remain scarce. Depending on the defect type that needs to be recognized, our task combines texture anomalies such as efflorescence or cracks with objects such as exposed reinforcement bars. Domain specific dataset contributions were very recently proposed with the "CrackForest" dataset [31], the CSSC database [36] and SDNET2018 [26]. However, as all of the former works feature a single-target and in fact single-class task, we have decided to extend existing work with the multi-class multi-target CODEBRIM dataset.

Defect (crack) recognition. Koch *et al.* [20] provide a comprehensive review on the state of computer vision in concrete defect detection and open aspects. In summary, the majority of approaches follow task specific modelling. Data-driven applications are still the exception and are yet to be leveraged fully. Recent works [23, 8, 18] show application to crack versus non-crack classification using images with little clutter and lack of structural context. An additional defect class of spalling is considered by the authors of [36]. Similar to other works, they focus on the single-target scenario and evaluation of well-known CNN baselines from prior object recognition literature. We extend their work by meta-learning more task specific neural architectures for more defect categories and overlapping defects.

Convolutional neural networks. A broader review of deep learning, its history and neural architecture innovations is given by LeCun *et al.* [22]. We recall some CNN architectures that serve as baselines and give a frame of reference for architectures produced by meta-learning on our task. Alexnet [21] had a large success on the ImageNet [29] challenge that was later followed by a set of deeper architectures commonly referred to as VGG [32]. Texture-CNN [1] is an adapted version of the Alexnet design that includes an energy-based adaptive feature pooling and FV-CNN [7] augments VGG with Fisher Vector pooling for texture classification. Recent works address information flow in deeper networks by adding skip connections with residual networks [14], wide residual networks (WRN) [37] and densely connected networks (DenseNet) [16].

Meta-learning neural architectures. Although deep neural networks empirically work well in many practical applications, networks have initially been designed for different tasks. A recent trend to bypass the human design intuition is to treat neural architectures themselves from a meta-learning perspective and conduct a black-box optimization on top of the training of weights to find suitable task-specific architecture designs. Several works in the literature have posed architecture meta-learning from a variety of perspectives based on reinforcement learning (RL) controllers [2, 38, 27, 4], differentiable methods [24] or evolutionary strategies [28]. In our work, we evaluate and adapt two RL based approaches to multi-target defect classification: MetaQNN [2] and "efficient neural architecture search" (ENAS) [27]. We pick these two approaches as they share underlying principles of training RL controllers. This allows us to pick a common reward metric determined by proposed CNN candidate accuracies. The main differences lie in the RL agents' nature: MetaQNN employs Q-Learning to learn to suggest increasingly accurate CNNs, whereas ENAS uses policy gradients [34] to train an auto-regressive recurrent neural network that samples individual layers based on previous input.

(a) Top row from left to right: 1.) exposed bars, spallation, cracks (hard to see) 2.) hairline crack with efflorescence 3.) efflorescence 4.) defect-free concrete. Bottom row from left to right: 1.) large spalled area with exposed bars and corrosion 2.) crack with graffiti 3.) corrosion stain, minor onset efflorescence 4.) defect-free concrete with dirt and markings.

(b) From left to right: 1.) spalled area with exposed bar, advanced corrosion and efflorescence 2.) exposed corroded bar 3.) larger crack 4.) partially exposed corroded bars, cracks 5.) hairline crack 6.) heavy spallation, exposed bars, corrosion 7.) wet/damp crack with efflorescence on the top 8.) efflorescence 9.) spalled area 10.) hairline crack with efflorescence.

Figure 1: Dataset examples. Top figure: full high-resolution images. Images heavily down-sampled for view in pdf. Bottom figure: Image patches cropped from annotated bounding boxes (not corresponding to top images). Images resized for view in pdf but with original aspect ratio.

3. The CODEBRIM dataset

The acquisition of the COncrete DEfect BRidge IMage: CODEBRIM dataset was driven by the need for a more diverse set of the often overlapping defect classes in contrast to previous crack focused work [31, 36, 26]. In particular, deep learning application to a real-world inspection scenario requires sampling of real-world context due to the many factors of variation in visual defect appearance. Our dataset is composed of five common defect categories: crack, spallation, exposed reinforcement bar, efflorescence (calcium leaching), corrosion (stains), found in 30 unique bridges (disregarding bridges that did not have defects). The bridges were chosen according to varying overall deterioration, defect extent, severity and surface appearance (e.g. roughness and color). Images were taken under changing weather conditions to include wet/stained surfaces with multiple cameras at varying scales. As most defects tend to be very small one crucial requirement was the acquisition at high-resolution. Considering that large parts of bridges are not accessible for a human, a subset of our dataset was acquired by UAV. We continue with the requirements and rationale behind the camera choices, the annotation process that led to the dataset and finally give a summary of important dataset properties.

3.1. Image acquisition and camera choice

Image acquisition and camera choices were motivated by typical concrete cracks in bridges having widths as small as $0.3\,\mathrm{mm}$ [20]. Resolving such defects on a pixel level

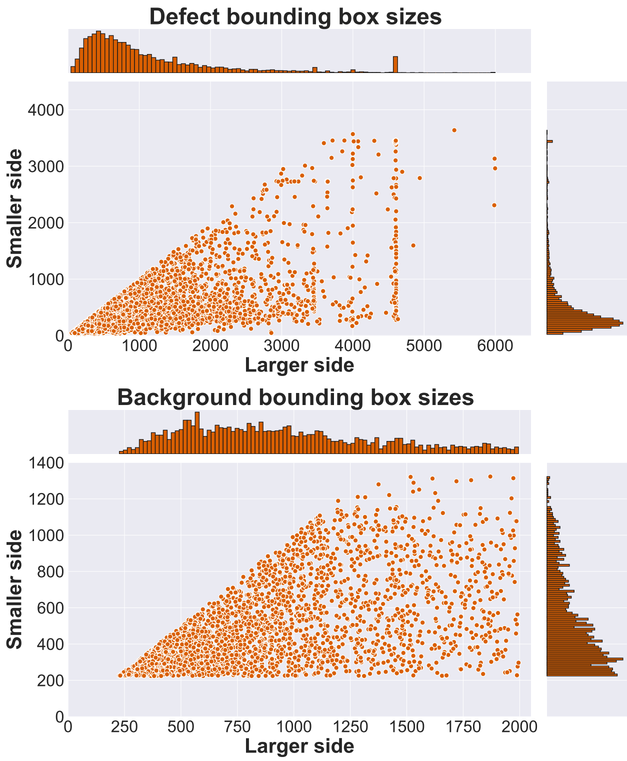

Figure 2: Top panel: distribution of annotated bounding box sizes for defects. Bottom panel: distribution of sizes for sampled non-overlapping background bounding boxes.

imposes a strong constraint on the distance and resolution at which the images are acquired. In a naive calculation for a conventional consumer-grade camera with an example chip of size $23.50 \times 15.60\,\mathrm{mm}$ and maximum resolution 6000×4000, this translates to around $0.1\,\mathrm{mm}$ per pixel at a focal length of $50\,\mathrm{mm}$ and a distance of roughly $1.5\,\mathrm{m}$ (assuming a pinhole camera model and viewing axis perpendicular to the surface). Based on this requirement our dataset was gathered with four different cameras at high resolution and large focal lengths under varying distance and angles. In addition, to homogeneously illuminate the darker bridge areas, we made use of diffused flash. Exact camera models and corresponding detailed parameters can be found in the supplementary material.

3.2. Dataset properties

We employed a multi-stage annotation process by first curating acquired images, annotating bounding boxes per defect and sequentially labelling each class separately. The rationale and exact annotation process is outlined in the supplementary material. The acquisition and annotation process resulted in a dataset with the following properties:

- 1590 high-resolution images with defects in context of 30 unique bridges, acquired at different scales and resolutions.
- 5354 annotated defect bounding boxes (largely with overlapping defects) and 2506 generated non-overlapping background bounding boxes.
- Defect numbers for the following classes: crack - 2507, spallation - 1898, efflorescence - 833, exposed bars - 1507 and corrosion stain - 1559.

Examples of images and extracted patches from bounding boxes featuring a variety of overlapping and non-overlapping defects can be seen in figure 1a and 1b respectively. We point out that in contrast to most object and texture based benchmarks, the majority of our dataset has more than one class occurring at once. We show a corresponding histogram for the number of defect classes per individual bounding box annotation in the supplementary material.

Apart from the multi-target nature making our dataset more challenging than single-class recognition, the task is difficult because of large variations in aspect ratio, scale and resolution of the different defects and their bounding boxes. This is true especially at a scene level, considering that cracks can be very fine and elongated, whereas spalled areas can vary almost arbitrarily. To illustrate these variations we visualize the distributions of defect bounding box sizes and the sampled background bounding box sizes in figure 2. Further details about distributions of image sizes, bounding box size distributions per category (with overlaps due to the multi-target nature) and distribution of aspect ratios per defect can be found in the supplementary material.

4. Meta-learning convolutional neural networks for multi-target defect classification

We use meta-learning to discover models tailored to multi-target defect classification on the CODEBRIM dataset. In order to find a suitable set of hyper-parameters for the meta-learning search space and training of neural architectures we start with the T-CNN [1] and VGG-A [32] baselines and investigate the influence of learning rate, batch size and patch size. For this we separate the dataset into train and validation splits and set aside a final test set for evaluation. We then adapt the MetaQNN [2] and ENAS [27] architecture meta-learning approaches and contrast the obtained results with the following set of CNN architectures proposed in the literature: Alexnet [21], T-CNN [1], VGG-A and VGG-D [32], wide residual network (WRN) [37] and densely connected convolutional networks (DenseNet) [16]. We want to point out that even though bounding box annotations are present in our dataset, we do not evaluate any bounding box detection algorithms because our goal at this stage is the establishment of the already challenging multi-target classification task. We have also evaluated

Architecture	Batch size	Multi-target accuracy [%] depending on learning rate schedule: max to min $\left[10^{-1}, 10^{-5}\right]$ best val	bv-test	bv-train	$\left[5 \cdot 10^{-2}, 5 \cdot 10^{-4}\right]$ best val	bv-test	bv-train	$\left[10^{-2}, 10^{-5}\right]$ best val	bv-test	bv-train
T-CNN	16	64.62	69.51	80.27	63.67	65.71	83.38	64.30	67.93	93.91
	32	64.78	66.19	87.66	63.36	68.72	94.49	62.84	66.35	96.22
	64	63.36	70.14	95.21	63.52	67.93	98.10	62.26	66.82	95.85
	128	63.67	67.45	98.31	63.36	66.82	98.63	60.53	65.08	94.47
VGG-A	16	60.22	62.08	75.74	63.67	68.24	94.78	64.93	70.45	98.29
	32	63.05	67.77	93.88	63.05	66.35	94.27	65.40	69.51	97.01
	64	63.36	69.66	98.00	63.37	70.45	90.64	59.90	63.82	97.01
	128	63.20	61.29	92.99	63.52	68.07	98.55	58.80	61.29	92.99

Table 1: Grid-search conducted on different batch sizes and different learning rate schedules for the T-CNN and VGG-A models. The multi-target best validation accuracy (best val) is shown together with each model's accuracy on the test set at the point in time of achieving the best validation accuracy (bv-test). The analogous training accuracy (bv-train) is shown to demonstrate that models do not under-fit. These validation accuracies have been used to determine training hyper-parameters.

transfer-learning from the ImageNet and MINC datasets, albeit without improvements and therefore report these experiments in the supplementary material.

4.1. Dataset training, validation and test splits

We have randomly chosen 150 unique defect examples per class for validation and test sets respectively. To avoid over-fitting due to very similar context, we make sure that we always include all annotated bounding boxes from one image in one part of the dataset split only. An alternative way to split the dataset is to separate train, validation and test sets according to unique bridges. However, it is infeasible to balance such a split with respect to equal amount of occurrences per defect due to individual bridges not featuring defect classes uniformly (particularly with class overlaps) and thus makes an unbiased training and reporting of average losses or accuracies difficult. Nevertheless, to investigate the importance of over-fitting global properties, we investigate and further discuss the challenges of such splits in the supplementary material.

4.2. Training procedure

The challenging multi-class multi-target nature of our dataset makes the following measures necessary:

1. **Multi-class multi-target.** For a precise estimate of a model's performance in a multi-target scenario, a classification is considered as correct if, and only if, all the targets are predicted correctly. To adapt all neural networks for this scenario we use a Sigmoid function for every class in conjunction with the binary cross entropy loss function. When we calculate classification accuracies we binarize the Sigmoid output with a threshold of 0.5. Note that this could be treated as a hyper-parameter to potentially obtain better results.
2. **Variations in scale and resolution.** We address the variation in scale and resolution of bounding boxes by following the common literature approach based on previous datasets such as ImageNet [29] and the models presented in [21, 32, 37, 16]. Here, the smaller side of the extracted patch is rescaled to a pre-determined patch size and random quadratic crops of patch size are taken to extract fixed size images during training.
3. **Train set imbalance.** We balance the training dataset by virtually replicating the under-represented class examples such that the overall defect number per class is on the same scale to make sure defect classes are sampled equally during training. Note that test and validation sets are balanced by design.

The reason for adopting step two is to allow for a direct comparison with CNNs proposed in prior literature without making modifications to their architectures. We do not use individual class accuracies as a performance metric as it is difficult to compare models that don't capture overlaps adequately. Nevertheless we provide an example table with multi-target versus per-class accuracy of later shown CNN literature baselines in the supplementary material.

4.2.1 Common hyper-parameters

We conduct an initial grid-search to find a suitable common set of hyper-parameters for CNNs (meta-learned or not) trained with stochastic gradient descent based on the T-CNN [1] and VGG-A [32] architectures. For this we use learning rate schedules with warm restarts (SGDWR) according to the work of [25]. The grid search features three cycles with ranges inspired by previous work [25, 27]: $\left[10^{-1}, 10^{-5}\right]$, $\left[5 \cdot 10^{-2}, 5 \cdot 10^{-4}\right]$ and $\left[10^{-2}, 10^{-5}\right]$, a warm restart cycle length of 10 epochs that is doubled after every restart, and four different batch sizes: $128, 64, 32$ and 16. All networks are trained for four warm restart cycles and thus 150 overall epochs after which we have noticed convergence. Other hyper-parameters are a momentum value of 0.9, a batch-normalization [17] value of 10^{-4} to accelerate training and a dropout rate [33] of 0.5 in the penultimate classification layer. Weights are initialized according to the Kaiming-normal distribution [13].

We determine a suitable set of hyper-parameters using cross-validation, that is according to the best validation accuracy during the entire training. We then report the test accuracy based on this model. We show the multi-target accuracy's dependency on learning rate and batch size for the two CNN architectures in table 1. We notice that the general trend is in favor of lower batch sizes and a learning rate schedule in the range of $\left[10^{-2}, 10^{-5}\right]$. While the evaluated best validation model's test accuracy generally follows a similar trend, the best test accuracies aren't always correlated with a higher validation accuracy, showing a light distribution mismatch between the splits. We further note that the absolute best test accuracy doesn't necessarily coincide with the point of training at which the model achieves the best validation accuracy. In general, the models seem to have a marginally higher accuracy for the test split. The table also shows that validation and test sets are reasonably different from the train set, on which all investigated models achieve an over-fit.

After determining a suitable set of hyper-parameters, a batch size of 16 and a learning rate cycle between $\left[10^{-2}, 10^{-5}\right]$, we have proceeded with the selection of patch sizes determined through an additional experiment based on best multi-target validation accuracy. We again emphasize that we do not pick hyper-parameters based on test accuracy, even if a model with lower validation accuracy has a better test score.

4.2.2 Selection of patch size

Whereas most CNN architectures proposed in the literature are designed for patch sizes of 224×224, we also evaluate a range of different patch sizes by modifying the number of parameters in the T-CNN model's first fully-connected layer according to the last convolution's spatial output resolution (we do not modify the outgoing feature amounts). In figure 3 we show the multi-target best validation and corresponding test accuracies for different patch and batch sizes. The perceivable trend is that models trained on patch sizes smaller than 224 yield less accuracy, whereas the validation accuracy seems to plateau or feature an upwards trend for larger patch sizes. The corresponding test accuracies mirror this trend. We leave the evaluation of even larger patch sizes for future work. For the remainder of this work, we continue

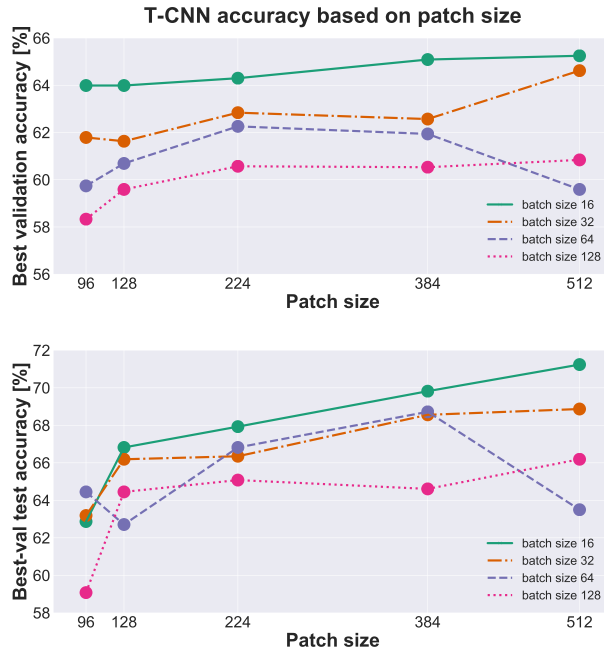

Figure 3: T-CNN multi-target validation accuracy (top panel) and best validation model's multi-target test accuracy (bottom panel) in dependence on patch size.

to use a patch size of 224. Although larger patch sizes seem promising they prevent a direct comparison and contrasting of meta-learning approaches with neural network models proposed in the literature without making modifications to their architectures.

4.2.3 Meta-learning specific parameters

We design the reward for both MetaQNN and ENAS to fit our multi-target scenario by setting it to the multi-target validation accuracy. We re-iterate that using a per-class accuracy as a metric and particularly to design an RL reward, could lead to controllers being biased towards naively raising the reward by generating models that predict (the easiest) subsets of classes correctly without considering the multi-target overlap properly. We try to set the method specific hyper-parameters of the two meta-learning methods as similar as possible to allow for a direct comparison. We therefore train all child CNN models using the SGDWR schedules and SGD hyper-parameters specified earlier.

MetaQNN: We employ an ϵ-greedy schedule for the Q-learning approach. We train an overall amount of 200 architectures and start with a full exploration phase of 100 architectures for $\epsilon = 1.0$. We continue with 10 architectures for ϵ values of 0.9 to 0.3 in steps of 0.1 and finish with 15 architectures for ϵ values of 0.2 and 0.1. Our search space is designed to allow neural architectures with at least 3 and a maximum of 10 convolutional layers. We include choices for quadratic filters in the sizes of $3, 5, 7, 9, 11$ with possible number of features per layer of $32, 64, 128, 256$. We use a Q-learning rate of 0.1, a discount-factor of 1.0 and an initial Q-value of 0.15. The latter is motivated by a 15% validation accuracy early-stopping criterion at the end of the first SGDWR cycle. In analogy to [2], if an architecture doesn't pass this threshold, it is discarded and a new one is sampled and trained.

Apart from the different reward design, we also make several extensions to the MetaQNN [2]: We cover down-sampling with an option for convolution stride $s = 2$ for filter sizes larger than 5. Convolutional layers are further followed by an adaptive pooling stage using spatial-pyramidal pooling (SPP) [12] of allowed scales $3, 4, 5$ and the possibility to pick a hidden fully-connected layer with size $32, 64$ or 128 before adding the final classification stage. All layers are followed by batch-normalization and a ReLU non-linearity to accelerate training. We also include the possibility to add ResNet-like skip connections between two padded 3×3 convolutions that do not change spatial dimensionality. If the number of convolutional output features is the same the skip connection is a simple addition, whereas an extra parallel convolution (that isn't counted as an additional layer) is added if the amount of output features needs to change. We make these extensions to provide a fairer comparison to the architecture search of ENAS, that by design contains batch-normalization, adaptive pooling and the possibility of adding skip-connections.

ENAS: In contrast to MetaQNN where the number of layers of each architecture is flexible, network depth in ENAS is pre-determined by the specification of number of nodes in the directed acyclic graph (DAG). Each node defines a possible set of feature operations that the RNN controller samples at each step together with connection patterns. In the process of the search, the same DAG is used to generate architectures with candidates sharing weights through sharing of feature operations. We choose to let the search evolve through alternate training of the CNNs' shared weights on the CODEBRIM train set and the RNN controller's weights on the validation set, where the controller samples one architecture per mini-batch. We design the DAG such that each architecture has 7 convolutional layers and 1 classification layer that is followed by a Sigmoid function. We choose this depth to have a direct comparison to the average depth of MetaQNN architectures. The allowed feature operations are convolutions with square filters of size 3 and 5, corresponding depth-wise separable convolutions [6], max-pooling and average-pooling with kernel size 3×3. Each layer is followed by batch-normalization and a ReLU non-linearity. Because ENAS uses shared weights in the search,

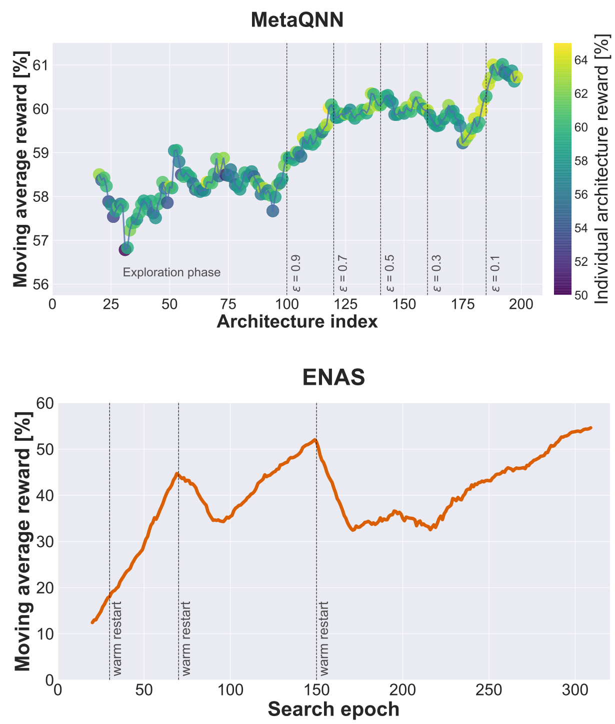

Figure 4: Evolution of the moving average reward defined as the multi-target validation accuracy of architectures proposed through meta-learning. The top panel additionally shows individual architecture accuracies for the MetaQNN in color. ENAS in the bottom panel has shared model weights during training and thus requires a final end-to-end re-training step for final validation accuracies of individual architectures.

a final re-training step of proposed architectures is necessary. We use a feature amount of 64 during the search for all layers and use a DenseNet growth-pattern [16] of $k = 2$ in the final training consistent with the work of Pham *et al.* [27]. The total number of search epochs is 310 (5 SGDWR cycles) after which we have experienced convergence of the controller. The RNN controller consists of an LSTM [15] with two hidden-layers of 64 features that is trained with a learning rate of 10^{-3} using ADAM [19].

4.3. Results and discussion

We demonstrate the effectiveness of neural architecture search with MetaQNN and ENAS for multi-target concrete defect classification on the CODEBRIM dataset. We show respective moving average rewards based on a window size of 20 architectures in figure 4. Individual architecture accuracies for MetaQNN are shown in color for each step in the top panel. We observe that after the initial exploration phase, the Q-learner starts to exploit and architec-

Architecture	Multi-target accuracy [%]		Params [M]	Layers
	best val	bv-test		
Alexnet	63.05	66.98	57.02	8
T-CNN	64.30	67.93	58.60	8
VGG-A	64.93	70.45	128.79	11
VGG-D	64.00	70.61	134.28	16
WRN-28-4	52.51	57.19	5.84	28
Densenet-121	65.56	70.77	11.50	121
ENAS-1	65.47	70.78	3.41	8
ENAS-2	64.53	68.91	2.71	8
ENAS-3	64.38	68.75	1.70	8
MetaQNN-1	66.02	68.56	4.53	6
MetaQNN-2	65.20	67.45	1.22	8
MetaQNN-3	64.93	72.19	2.88	7

Table 2: Comparison of popular CNNs from the literature with the top three architectures of MetaQNN and ENAS in terms of best multi-target validation accuracy (best val), best validation model's test accuracies (bv-test), overall amount of parameters (Params) in million and amount of trainable layers. For WRN we use a width factor of 4 and a growth rate of $k = 32$ for DenseNet.

tures improve in multi-target validation accuracy. In the bottom panel of the figure we show corresponding rewards for the shared-weight ENAS DAG. We observe that both methods learn to suggest architectures with improved accuracy over time. We remind the reader that in contrast to the MetaQNN, a final re-training step of the top architectures is needed for ENAS to obtain the task's final accuracy values.

The multi-target validation and test accuracies, again reported at the point in time of best validation, the number of overall architecture parameters and layers for the top three MetaQNN and ENAS architectures can be found in table 2. We also evaluate and provide these values for popular CNN baselines: Alexnet [21], VGG [32], Texture-CNN [1], wide residual networks (WRN) [37] and densely connected networks (DenseNet) [16]. We see that the Texture-CNN variant of Alexnet slightly outperforms the latter. The connectivity pattern of the DenseNet architecture also boosts the performance in contrast to the VGG models. Lastly, we note that we were only able to achieve heavy over-fitting with WRN configurations (even with other hyper-parameters and other configurations such as WRN-28-10 or WRN-40).

The accuracies obtained by all of our meta-learned architectures, independently of the underlying algorithm, outperform most baseline CNNs and feature at least similar performance in comparison to DenseNet. Moreover, they feature much fewer parameters with fewer overall layers and are thus more efficient than their computationally heavy counterparts. Our best meta-learned models have validation accuracies as high as 66%, while the test accuracies go up to 72% with total amount of parameters less than 5 million. In contrast to literature CNN baselines these architectures are thus more tailored to our specific task and its multi-target nature. Interestingly, previously obtained improvements from one literature CNN baseline to another on ImageNet, such as Alexnet 81.8% to VGG-D 92.8% top-5 accuracies, do not show similar improvements when evaluated on our task. This underlines the need for diverse datasets in evaluation of architectural advances and demonstrates how architectures that were hand-designed, even with incredible care and effort, for one dataset such as ImageNet may nonetheless be inferior to meta-learned neural networks.

Between the two search strategies we do not observe a significant difference in performances. We believe this is due to previously mentioned modifications to MetaQNN, mainly the addition of skip-connections and batch-normalization that make proposed architectures more similar to those of ENAS. We point the reader to the supplementary material for exact definitions of meta-learned architectures. There, we also include a set of image patches that are commonly classified as correct for all targets, images where only part of the overlapping defect classes is predicted and completely misclassified examples.

5. Conclusion

We introduce a novel multi-class multi-target dataset called CODEBRIM for the task of concrete defect recognition. In contrast to previous work that focuses largely on cracks, we classify five commonly occurring and structurally relevant defects through deep learning. Instead of limiting our evaluation to common CNN models from the literature, we adapt and compare two recent meta-learning approaches to identify suitable task-specific neural architectures. Through extension of the MetaQNN, we observe that the two meta-learning techniques yield comparable architectures. We show that these architectures feature fewer parameters, fewer layers and are more accurate than their human designed counterparts on our presented multi-target classification task. Our best meta-learned models achieve multi-target test accuracies as high as 72%. Our work creates prospects for future work such as multi-class multi-target concrete defect detection, semantic segmentation and system applications like UAV based real-time inspection of concrete structures.

Acknowledgements: This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 687384 "AEROBI". We would like to thank everyone involved in the AEROBI project. Particular appreciation is given for the civil engineering team of Egnatia Odos A.E. and Netivei NTIC, without whom the annotation of the dataset wouldn't have been possible. We further thank FADA-CATEC, Tobias Weis and Sumit Pai for their support in parts of the data acquisition and Hieu Pham for valuable discussion of ENAS hyper-parameters.

References

[1] Vincent Andrearczyk and Paul F Whelan. Using filter banks in Convolutional Neural Networks for texture classification. *Pattern Recognition Letters*, 84:63–69, 2016. 1, 2, 4, 5, 8

[2] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing Neural Network Architectures using Reinforcement Learning. *International Conference on Learning Representations (ICLR)*, 2016. 2, 4, 7

[3] Sean Bell, Paul Upchurch, Noah Snavely, and Kavita Bala. Material recognition in the wild with the Materials in Context Database. In *Computer Vision and Pattern Recognition (CVPR)*, 2015. 1, 2

[4] Han Cai, Tianyao Chen, Weinan Zhang, Yong Yu, and Jun Wang. Efficient Architecture Search by Network Transformation. *AAAI Conference on Artificial Intelligence (AAAI)*, 2018. 2

[5] Barbara Caputo, Eric Hayman, and P Mallikarjuna. Class-specific material categorisation. In *International Conference on Computer Vision (ICCV)*, 2005. 2

[6] Francois Chollet. Xception: Deep Learning with Depthwise Seperable Convolutions. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1800–1807, 2017. 7

[7] Mircea Cimpoi, Subhransu Maji, and Andrea Vedaldi. Deep convolutional filter banks for texture recognition and segmentation. In *Computer Vision and Pattern Recognition (CVPR)*, 2015. 2

[8] Wilson R. L. da Silva and Diogo S. de Lucena. Concrete Cracks Detection Based on Deep Learning Image Classification. In *International Conference on Experimental Mechanics (ICEM18)*, 2018. 1, 2

[9] Kristin J. Dana, Bram van Ginneken, Shree K. Nayar, and Jan J. Koenderink. Reflectance and texture of real-world surfaces. *ACM Transactions on Graphics (TOG)*, 18(1):1–34, 1999. 2

[10] Mark Everingham, S. M.Ali Ali Eslami, Luc Van Gool, Christopher K.I. I Williams, John Winn, and Andrew Zisserman. The Pascal Visual Object Classes Challenge: A Retrospective. *International Journal of Computer Vision (IJCV)*, 111(1):98–136, 2014. 1, 2

[11] Eric Hayman, Barbara Caputo, Mario Fritz, and Jan-Olof Eklundh. On the Significance of Real-World Conditions for Material Classification. In *European Conference on Computer Vision (ECCV)*, 2004. 2

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. In *European Conference on Computer Vision (ECCV)*, pages 346–361, 2014. 7

[13] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *International Conference on Computer Vision (ICCV)*, pages 1026–1034, 2015. 6

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[15] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997. 7

[16] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269, 2017. 1, 2, 4, 5, 7, 8

[17] S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *International Conference on Machine Learning (ICML)*, volume 37, pages 448–456, 2015. 6

[18] Hyunjun Kim, Eunjong Ahn, Myoungsu Shin, and Sung-Han Sim. Crack and Noncrack Classification from Concrete Surface Images Using Machine Learning. *Structural Health Monitoring*, 2018. 1, 2

[19] Diederik P. Kingma and Jimmy Lei Ba. Adam: a Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*, 2015. 7

[20] Christian Koch, Kristina Georgieva, Varun Kasireddy, Burcu Akinci, and Paul Fieguth. A review on computer vision based defect detection and condition assessment of concrete and asphalt civil infrastructure. *Advanced Engineering Informatics*, 29(2):196–210, 2015. 2, 3

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Neural Information Processing Systems (NeurIPS)*, volume 25, pages 1097–1105, 2012. 1, 2, 4, 5, 8

[22] Yann Lecun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. 2

[23] Yundong Li, Hongguang Li, and Hongren Wang. Pixel-wise crack detection using deep local pattern predictor for robot application. *Sensors*, 18(9), 2018. 1, 2

[24] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable Architecture Search. *International Conference on Learning Representations (ICLR)*, 2019. 2

[25] I. Loshchilov and F. Hutter. SGDR: Stochastic Gradient Descent With Warm Restarts. In *International Conference on Learning Representations (ICLR)*, 2017. 5

[26] Marc Macquire, Sattar Dorafshan, and Robert J. Thomas. SDNET2018: A concrete crack image dataset for machine learning applications. *https://digitalcommons.usu.edu/all_datasets/48 (last access: 06.11.18)*, Paper 48, 2018. 1, 2, 3

[27] Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Efficient Neural Architecture Search via Parameters Sharing. In *International Conference on Machine Learning (ICML)*, 2018. 2, 4, 5, 7

[28] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Quoc Le, and Alex Kurakin. Large-Scale Evolution of Image Classifiers. In *International Conference on Machine Learning (ICML)*, 2017. 2

[29] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. In *International Journal of Computer Vision (IJCV)*, volume 115, pages 211–252, 2015. 1, 2, 5

[30] Lavanya Sharan, Ruth Rosenholtz, and Edward H. Adelson. Material perception: What can you see in a brief glance? *Journal of Vision (JOV)*, 9(8), 2009. 2

[31] Yong Shi, Limeng Cui, Zhiquan Qi, Fan Meng, and Zhensong Chen. Automatic road crack detection using random structured forests. *IEEE Transactions on Intelligent Transportation Systems (T-ITS)*, 17(12):3434–3445, 2016. 1, 2, 3

[32] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations (ICLR)*, 2015. 1, 2, 4, 5, 8

[33] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout : A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research (JMRL)*, 15:1929–1958, 2014. 6

[34] S. Richard Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *Neural Information Processing Systems (NeurIPS)*, pages 1057–1063, 1999. 2

[35] Jianxiong Xiao, James Hays, Krista A Ehinger, and Antonio Torralba. SUN Database : Large-scale Scene Recognition from Abbey to Zoo. In *Computer Vision and Pattern Recognition (CVPR)*, 2010. 1, 2

[36] Liang Yang, Bing Li, Wei Li, Zhaoming Liu, Guoyong Yang, and Jizhong Xiao. Deep Concrete Inspection Using Unmanned Aerial Vehicle Towards CSSC Database. In *International Conference on Intelligent Robots and Systems (IROS)*, 2017. 1, 2, 3

[37] Sergey Zagoruyko and Nikos Komodakis. Wide Residual Networks. In *British Machine Vision Conference (BMVC)*, pages 87.1–87.12, 2016. 1, 2, 4, 5, 8

[38] Barret Zoph and Quoc V. Le. Neural Architecture Search with Reinforcement Learning. In *International Conference on Learning Representations (ICLR)*, 2017. 2