A. Dataset

A.1. NAS-Bench-201

We base our evaluations on the NAS-Bench-201 [5] search space. It is a cell-based architecture search space. Each cell has in total 4 nodes and 6 edges. The nodes in this search space correspond to the architecture's feature maps and the edges represent the architectures operation, which are chosen from the operation set $\mathcal{O} = \{1 \times 1 \text{ conv.}, 3 \times 3 \text{ conv.}, 3 \times 3 \text{ avg. pooling}, \text{skip}, \text{zero}\}$ (see Figure 1). This search space contains in total $5^6 = 15\,625$ architectures, from which only 6 466 are unique, since the operations skip and zero can cause isomorphic cells (see Figure 6), where the latter operation zero stands for dropping the edge. Each architecture is trained on three different image datasets for 200 epochs: CIFAR-10 [10], CIFAR-100 [10] and ImageNet16-120 [3]. For our evaluations, we consider all unique architectures in the search space and test splits of the corresponding datasets. Hence, we evaluate $3 \cdot 6\,466 = 19\,398$ pretrained networks in total.

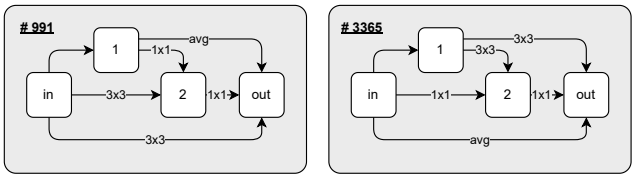

Figure 6. Example of two isomorphic graphs in NAS-Bench-201. Due to the skip connection from node `in` to node `1`, both computational graphs are equivalent, but their identification in the search space is different. For this dataset, we evaluated all non-isomorphic graphs (`#991` was evaluated and `#3365` was not).

A.2. Dataset Gathering

We collect evaluations for our dataset for different corruptions and adversarial attacks (as discussed in subsection 2.2 and subsection 2.3) following algorithm 1. This process is also depicted in Figure 7. Hyperparameter settings for adversarial attacks are listed in Table 2. Due to the heavy load of running all these evaluations, they are performed on several clusters. These clusters are comprised of either (i) compute nodes with Nvidia A100 GPUs, 512 GB RAM, and Intel Xeon IceLake-SP processors, (ii) compute nodes with NVIDIA Quadro RTX 8000 GPUs, 1024 GB RAM, and AMD EPYC 7502P processors, (iii) NVIDIA Tesla A100 GPUs, 2048 GB RAM, Intel Xeon Platinum 8360Y processors, and (iv) NVIDIA Tesla A40 GPUs, 2048 GB RAM, Intel Xeon Platinum 8360Y processors.

A.3. Dataset Structure, Distribution, and License

Files are provided in `json` format to ensure platform-independence and to reduce the dependency on external libraries (e.g. Python has built-in `json`-support).

Table 2. Hyperparameter settings of adversarial attacks evaluated.

Attack	Hyperparameters
FGSM	$\epsilon \in \{.1, .5., 1, 2, 3, 4, 5, 6, 7, 8, 255\}/255$
PGD	$\epsilon \in \{.1, .5., 1, 2, 3, 4, 8, 255\}/255$ $\alpha = 0.01/0.3$ 40 attack iterations
APGD	$\epsilon \in \{.1, .5., 1, 2, 3, 4, 8, 255\}/255$ 100 attack iterations
Square	$\epsilon \in \{.1, .5., 1, 2, 3, 4, 8, 255\}/255$ 5 000 search iterations

Algorithm 1: Robustness Dataset Gathering

Input: (i) Architecture space A (NAS-Bench-201).
Input: (ii) Test datasets D (CIFAR-10, CIFAR-100, ImageNet16-120).
Input: (iii) Set of attacks and/or corruptions C.
Input: (iv) Robustness Dataset R.

for $a \in A$ **do**
  ▷ `Load pretrained weights for` a`.`
  a.load_weights(d)
  for $d \in D$ **do**
    for $c(\cdot, \cdot) \in C$ **do**
      ▷ `Corrupt dataset` d`.`
      $d_c \leftarrow c(a, d)$
      ▷ `Evaluate architecture` a `with` d_c`.`
      Accuracy, Confidence, ConfusionMatrix $\leftarrow$ $eval(a, d_c)$
      ▷ `Extend robustness dataset with evaluations.`
      $R[d][c]$["accuracy"][a] $\leftarrow$ Accuracy
      $R[d][c]$["confidence"][a] $\leftarrow$ Confidence
      $R[d][c]$["cm"][a] $\leftarrow$ ConfusionMatrix
    end
  end
end

We will publish code that accompanies our dataset on GitHub. The dataset itself will be linked from GitHub and is hosted on an institutional cloud service. This ensures long-time availability and the possibility to version the dataset. Dataset and code will be published at the notification date under GNU GPLv3.

A.4. Structure

The dataset consists of 3 folders, one for each dataset evaluated (`cifar10`, `cifar100`, `ImageNet16-120`). Each folder contains one `json` file for each combination of

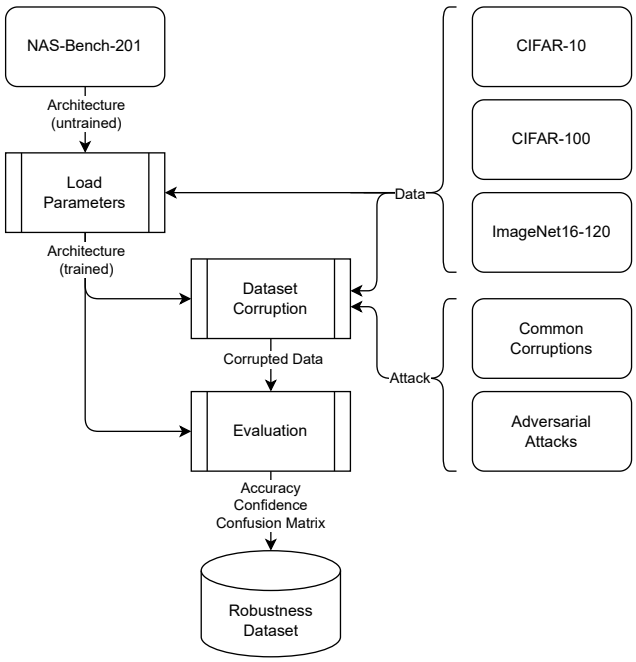

Figure 7. Diagram showing the gathering process for our robustness dataset. (i) An non-isomorphic architecture contained in NAS-Bench-201 is created and its parameters are loaded from a provided checkpoint, dependent on the dataset evaluated. (ii) Given the evaluation dataset, an attack or corruption, and the trained network, the evaluation dataset is corrupted and (iii) the resulting corrupted data is used to evaluate the network. (iv) The evaluation results are stored in our robustness dataset.

key and measurement. Keys refer to the sort of attack or corruption used (Table 3 lists all keys). Measurements refer to the collected evaluation type (`accuracy`, `confidence`, `cm`). Clean and adversarial evaluations are performed on all datasets, while common corruptions are evaluated on `cifar10` and `cifar100`. Additionally, the dataset contains one metadata file (`meta.json`).

Metadata The `meta.json` file contains information about each architecture in NAS-Bench-201. This includes, for each architecture identifier, the corresponding string defining the network design (as per [5]) as well as the identifier of the corresponding non-isomorphic architecture from [5] that we evaluated. The file also contains all ϵ values that we evaluated for each adversarial attack. An excerpt of this file is shown in Figure 8.

Files All files are named according to `"{key}_{measurement}.json"`. Hence, the path to all clean accuracies on `cifar10` is `"./cifar10/clean_accuracy.json"`. An excerpt of this file is shown in Figure 9. Each file contains nested dictionaries stating the dataset, evaluation key and measurement type. For evaluations with multiple measurements, e.g. in the case of adversarial attacks for multiple ϵ values, the results are concatenated into a list. Files and their possible contents are described in Table 4.

Table 3. Keys for attacks and corruptions evaluated.

Clean	Adversarial	Common Corruptions
clean	aa_apgd-ce	brightness
	aa_square	contrast
	fgsm	defocus_blur
	pgd	elastic_transform
		fog
		frost
		gaussian_noise
		glass_blur
		impulse_noise
		jpeg_compression
		motion_blur
		pixelate
		shot_noise
		snow
		zoom_blur


```
{
  "ids": {
    ...,
    "21": {
      "nb201-string": "|nor_conv_1x1~0|+|none~0|none~1|+{...}",
      "isomorph": "21"
    },
    ...,
    "1832": {
      "nb201-string": "|nor_conv_1x1~0|+|nor_conv_1x1~0|none~1|+{...}",
      "isomorph": "309"
    },
    ...
  },
  "epsilons": {
    "aa_apgd-ce": [0.1, 0.5, 1.0, 2.0, 3.0, 4.0, 8.0],
    "aa_square": [0.1, 0.5, 1.0, 2.0, 3.0, 4.0, 8.0],
    "fgsm": [0.1, 0.5, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 255.0],
    "pgd": [0.1, 0.5, 1.0, 2.0, 3.0, 4.0, 8.0]
  }
}
```


Figure 8. Excerpt of `meta.json` showing meta information of architectures #`21` and #`1832`, as well as ϵ values for each attack. Architecture #`21` is non-isomorphic and points to itself, while architecture #`1832` is an isomorphic instance of #`309`.

```
{
  "cifar10": {
    "clean": {
      "accuracy": {
        "0": 0.856,
        ...
      }
    }
  }
}
```

```
{
  "cifar10": {
    "pgd": {
      "accuracy": {
        "0": [0.812, 0.582, 0.295, 0.034, 0.002, 0.0, 0.0],
        ...
      }
    }
  }
}
```

Figure 9. Excerpt of **(left)** `clean_accuracy.json` and **(right)** `pgd_accuracy.json` for dataset `cifar10` for the architecture #`0`. Numbers are rounded to improve readability.

Table 4. Files and their possible content.

File	Description
clean_accuracy	one accuracy value for each evaluated network
clean_confidence	one confidence matrix for each evaluated network and collection scheme
clean_cm	one confusion matrix for each evaluated network
{attack}_accuracy	list of accuracies, where each element corresponds to the respective ϵ value
{attack}_confidence	list of confidence matrices, where each element corresponds to the respective ϵ value
{attack}_cm	list of confusion matrices, where each element corresponds to the respective ϵ value
{corruption}_accuracy	list of accuracies, where each element corresponds to the respective corruption severity
{corruption}_confidence	list of confidence matrices, where each element corresponds to the respective corruption severity
{corruption}_cm	list of confusion matrices, where each element corresponds to the respective corruption severity

We showed some analysis and possible use-cases on accuracies in the main paper. In the following, we elaborate on and show `confidence` and confusion matrix (`cm`) measurements.

A.5. Confidence

We collect the mean confidence after softmax for each network over the whole (attacked) test dataset evaluated. We used 3 schemes to collect confidences (see Figure 11). First, confidences for each class are given by true labels (called `label`). In case of `cifar10`, this results in a 10×10 confidence matrix, for `cifar100` a 100×100 confidence matrix, and `ImageNet16-120` a 120×120 confidence matrix. Second, confidences for each class are given by the class predicted by the network (called `argmax`). This again results in matrices of sizes as mentioned. Third, confidences for correctly classified images as well as confidences for incorrectly classified images (called `prediction`). For all image datasets, this results in a vector with 2 dimensions. Each result is saved as a list (or list of list), see Figure 10.

Figure 12 shows a progression of `label` confidence values for class label 0 on `cifar10` from `clean` to `fgsm` with increasing values of ϵ. Figure 13 shows how `prediction` confidences of correctly and incorrectly classified images correlate with increasing values of ϵ when attacked with `fgsm`.

```
{
  "cifar10": {
    "clean": {
      "confidence": {
        "0": {
          "label": [[...]],
          "argmax": [[...]],
          "prediction": [...]
        }
      }
    }
  }
}
```

Figure 10. Excerpt of `clean_confidence.json` for `cifar10`. Numbers are not shown to improve readability.

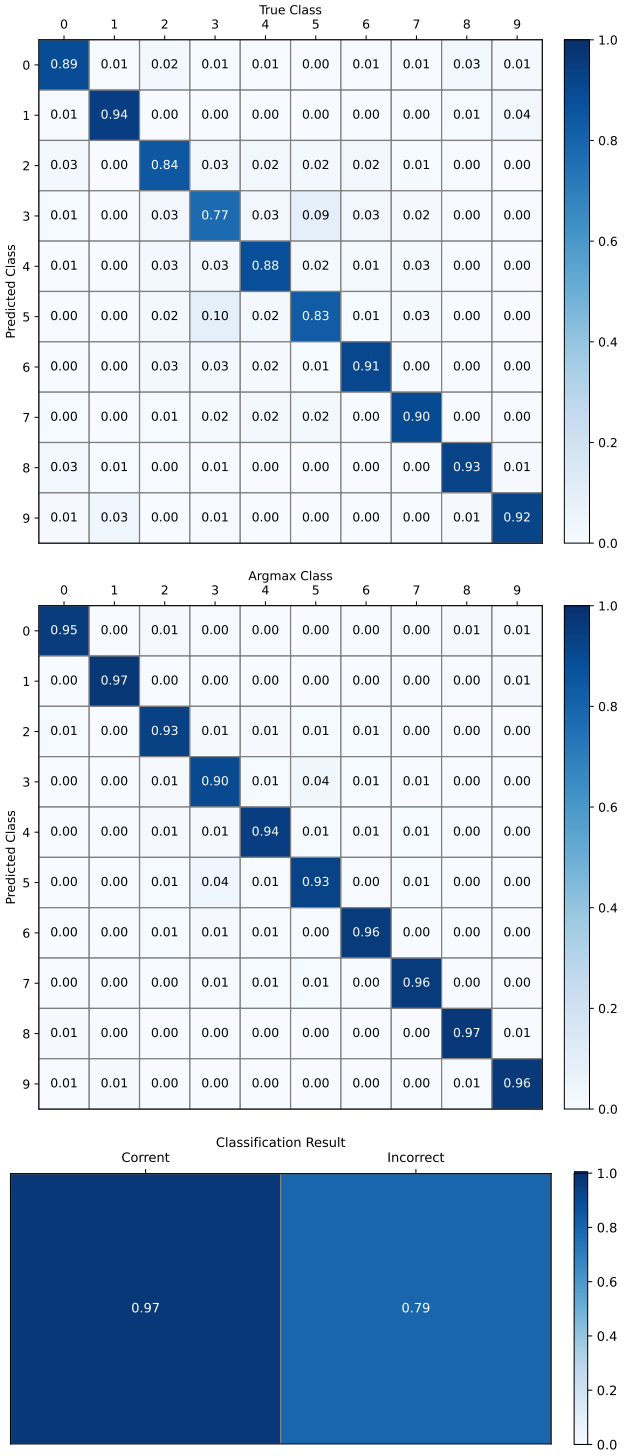

Figure 11. Mean confidence scores on clean CIFAR-10 images for all non-isomorphic networks in NAS-Bench-201. (**top**: `label`) For each true class label. (**middle**: `argmax`) For each predicted class label. (**bottom**: `prediction`) For correct and incorrect classifications.

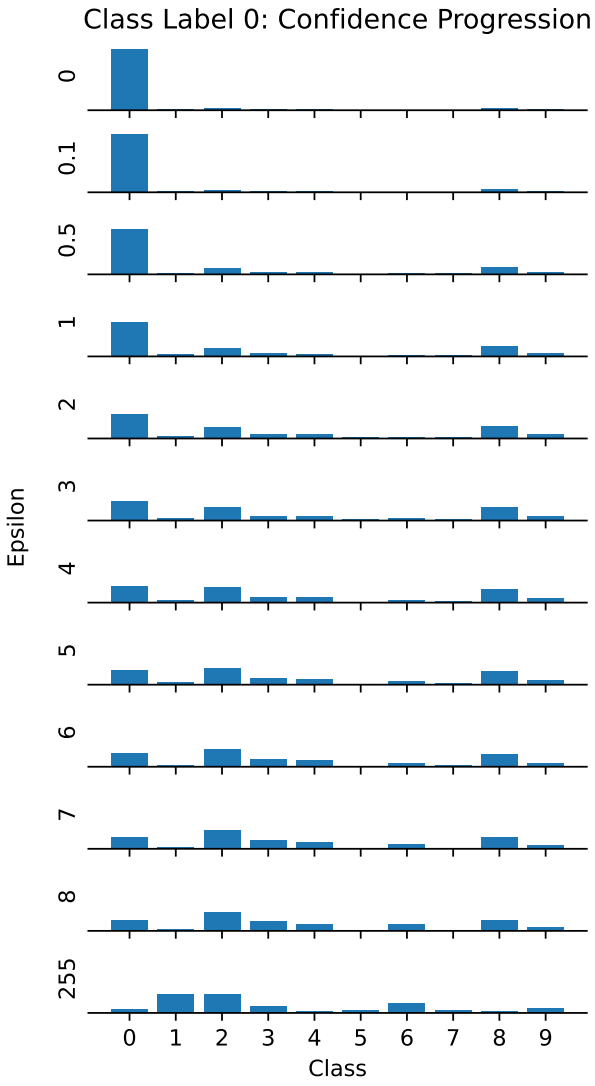

Figure 12. Mean `label` confidence scores on FGSM-attacked CIFAR-10 images for different ϵ for all non-isomorphic networks in NAS-Bench-201. Only confidence scores for class label 0 are shown. Networks lose prediction confidence for the true label when ϵ increases.

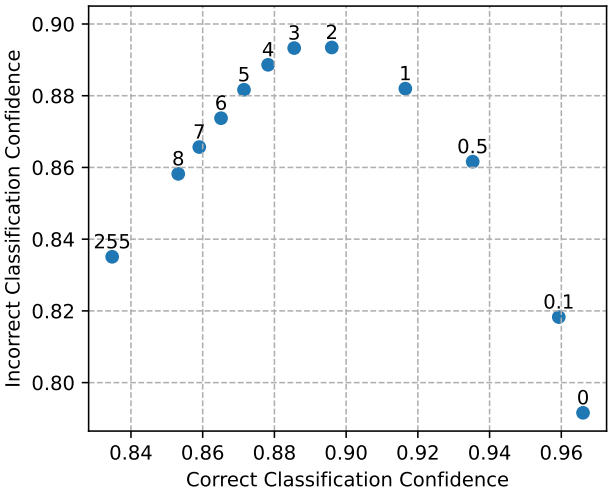

Figure 13. Mean `prediction` confidence scores on FGSM-attacked CIFAR-10 images for different ϵ (on top of points) for all non-isomorphic networks in NAS-Bench-201. Networks become less confident in their prediction if their prediction is correct when ϵ increases. Networks become more confident in their prediction if their prediction is incorrect, however, only up to a certain ϵ value. When ϵ further increases, confidence drops again.

A.6. Confusion Matrix

For each evaluated network, we collect the confusion matrix (key: `cm`) for the corresponding (attacked) test dataset. The result is a 10×10 matrix in case of `cifar10`, a 100×100 matrix in case of `cifar100`, and a 120×120 matrix in case of `ImageNet16-120`. See Figure 14 for an example, where we summed up confusion matrices for all networks on `cifar10`.

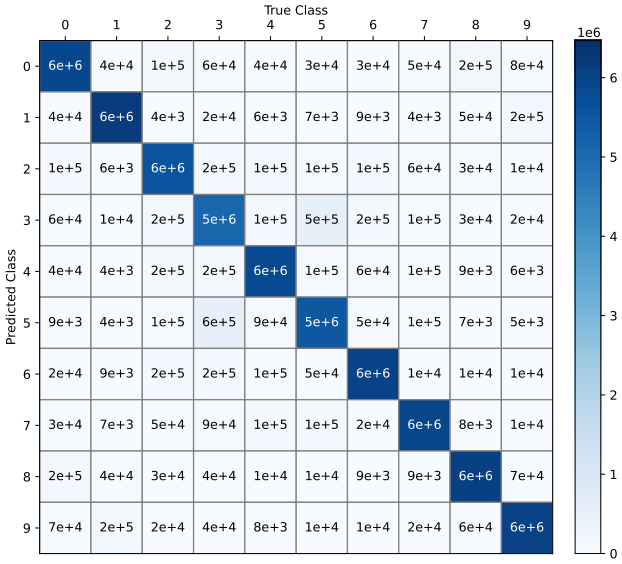

Figure 14. Aggregated confusion matrices on clean CIFAR-10 images for all non-isomorphic networks in NAS-Bench-201.

A.7. Correlations between Image Datasets

In Figure 15 we show the correlation between all clean and adversarial accuracies over all datasets collected. This plot shows a positive correlation between the image datasets for the one-step FGSM attack, whereas for all other multi-step attacks, the correlation becomes close to zero or even negative.

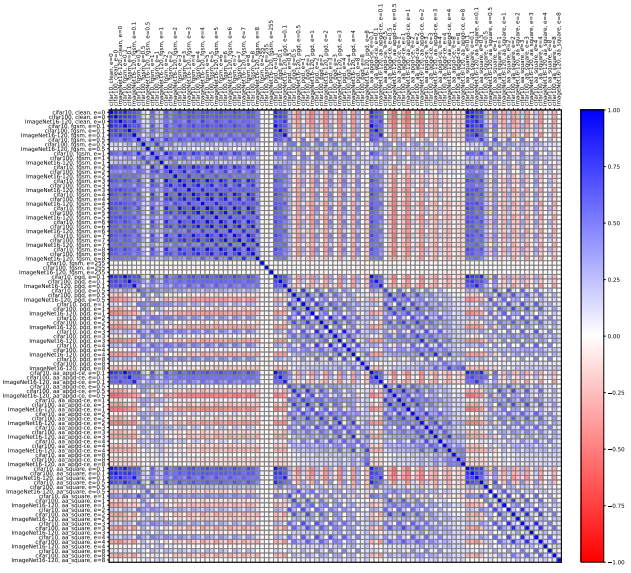

Figure 15. Kendall rank correlation coefficient between all clean and adversarial accuracies that are evaluated in our dataset.

A.8. Example image of corruptions in CIFAR-10-C

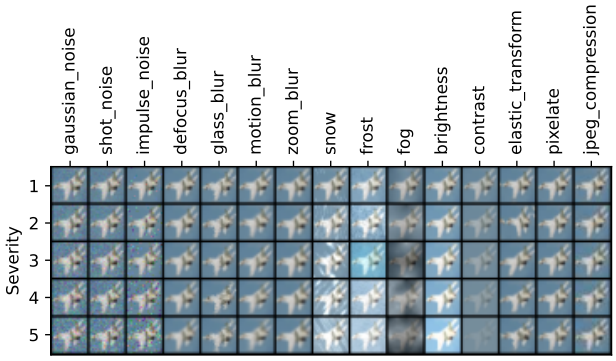

Figure 16. An example image of CIFAR-10-C [7] with different corruption types at different severity levels. CIFAR-100-C [7] consists of images with the same corruption types and severity levels.

A.9. Main Paper Figures for other Image Datasets

A.9.1 CIFAR-100 Adversarial Attack Accuracies (Figure 2)

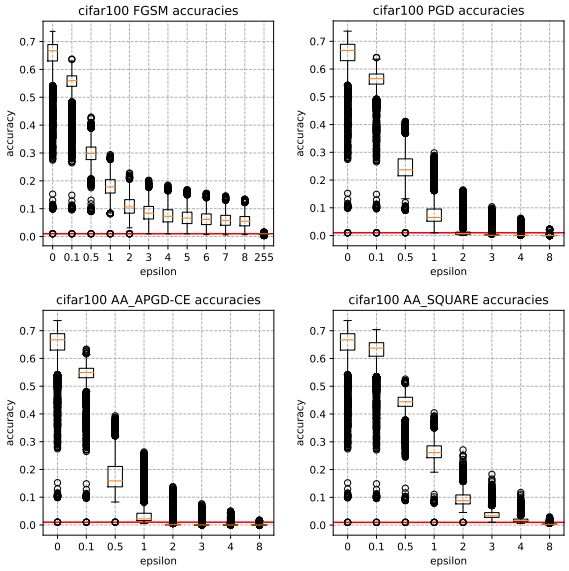

Figure 17. Accuracy boxplots over all unique architectures in NAS-Bench-201 for different adversarial attacks (FGSM [6], PGD [11], APGD [4], Square [1]) and perturbation magnitude values ϵ, evaluated on CIFAR-100. Red line corresponds to guessing.

A.9.2 ImageNet16-120 Adversarial Attack Accuracies (Figure 2)

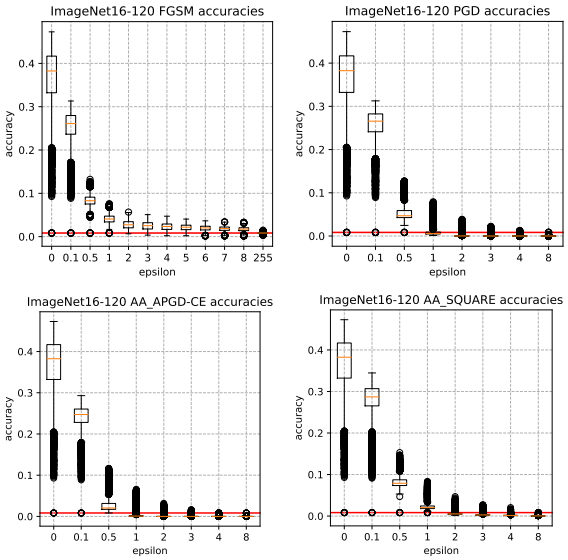

Figure 18. Accuracy boxplots over all unique architectures in NAS-Bench-201 for different adversarial attacks (FGSM [6], PGD [11], APGD [4], Square [1]) and perturbation magnitude values ϵ, evaluated on ImageNet16-120. Red line corresponds to guessing.

A.9.3 CIFAR-10-C Common Corruption Accuracies (Figure 2)

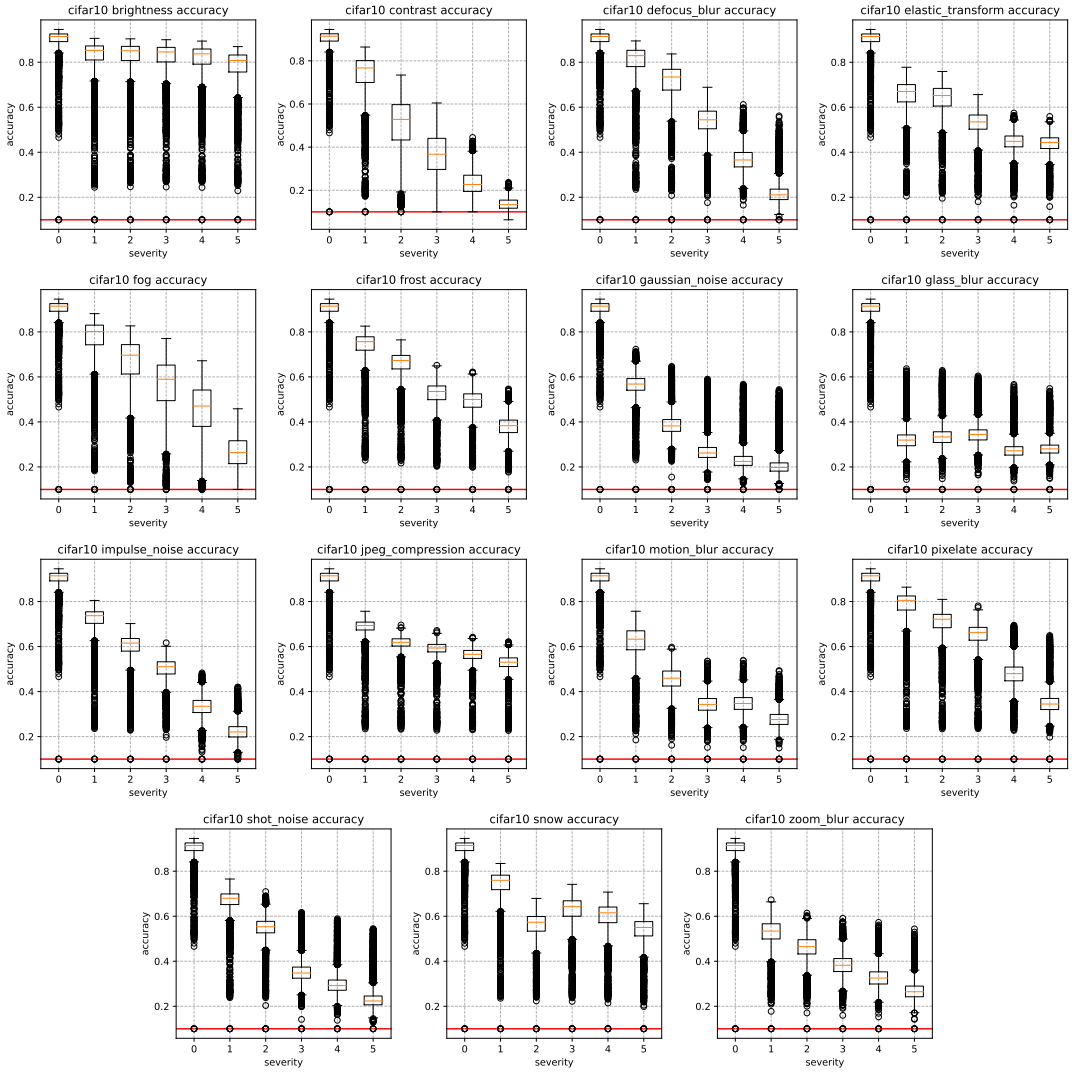

Figure 19. Accuracy boxplots over all unique architectures in NAS-Bench-201 for different corruption types at different severity levels, evaluated on CIFAR-10-C. Red line corresponds to guessing.

A.9.4 CIFAR-100-C Common Corruption Accuracies (Figure 2)

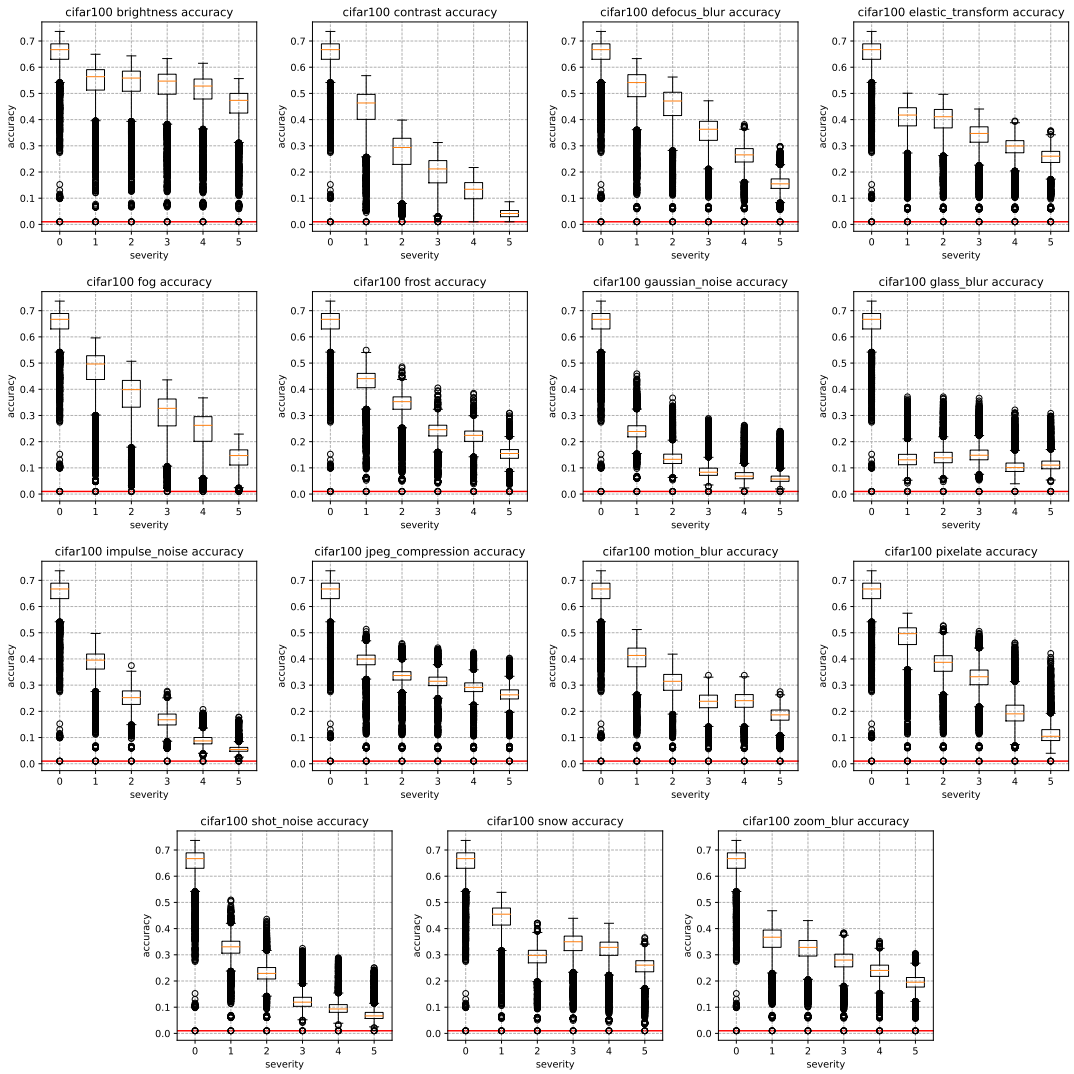

Figure 20. Accuracy boxplots over all unique architectures in NAS-Bench-201 for different corruption types at different severity levels, evaluated on CIFAR-100-C. Red line corresponds to guessing.

A.9.5 CIFAR-100 Adversarial Attack Correlations (Figure 3)

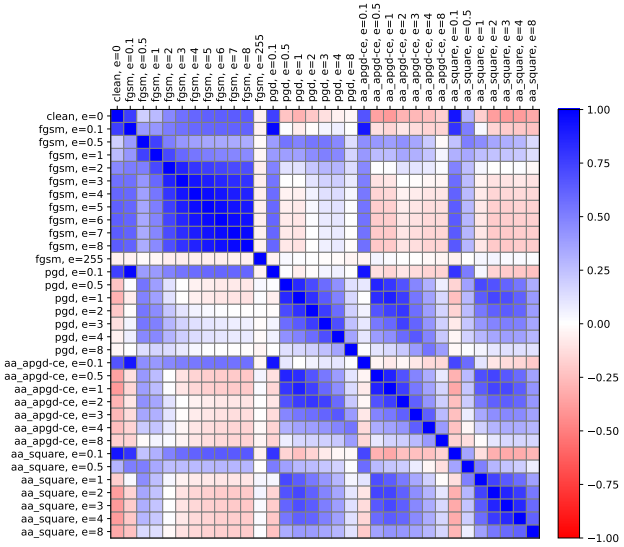

Figure 21. Kendall rank correlation coefficient between clean accuracies and robust accuracies on different attacks and magnitude values ϵ on CIFAR-100 for all unique architectures in NAS-Bench-201.

A.9.6 ImageNet16-120 Adversarial Attack Correlations (Figure 3)

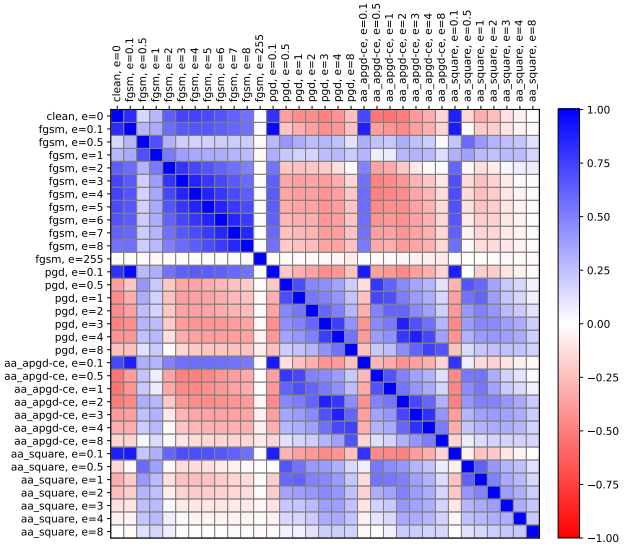

Figure 22. Kendall rank correlation coefficient between clean accuracies and robust accuracies on different attacks and magnitude values ϵ on ImageNet16-120 for all unique architectures in NAS-Bench-201.

A.9.7 CIFAR-100-C Common Corruption Correlations

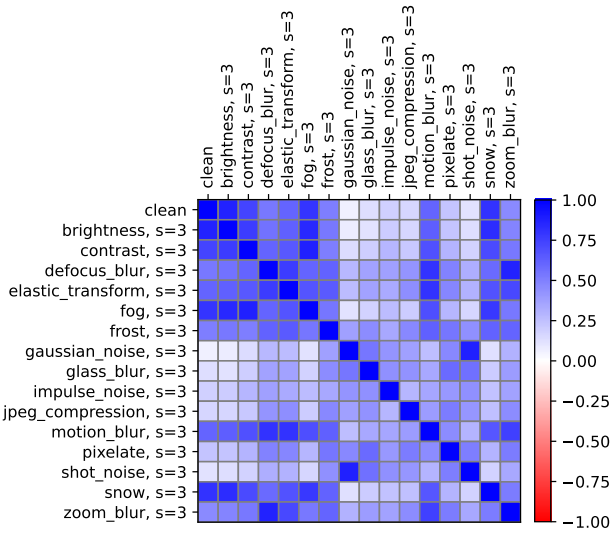

Figure 23. Kendall rank correlation coefficient between clean accuracies and accuracies on different corruptions at severity level 4 on CIFAR-100-C for all unique architectures in NAS-Bench-201.

B. Analysis

In this section, we first depict the best architectures in NAS-Bench-201 [5] in subsection B.1, then show the effect of parameter count on robustness and the magnitude of potential gains in robustness in a limited parameter count setting in subsection B.2, and lastly show the effect of single changes to the best performing architecture according to clean accuracy in subsection B.3.

B.1. Best Architectures

Figure 24 visualizes the best architectures in the NAS-Bench-201 [5] search space in terms of clean accuracy, mean adversarial accuracy, and mean common corruption accuracy on CIFAR-10 and their respective edit distances. The edit distance is defined by the number of changes, either node or edge, to change the graph to the target graph. In the case of NAS-Bench-201 architectures, an edit distance of 1 means that exactly one operation differs between two architectures. So in order to modify the best performing architecture in terms of clean accuracy (#13714) into the best performing architecture according to mean corruption accuracy (#3456), we need to exchange two (out of six) operations: (i) exchange operation 2 from 3×3 convolution to zero and (ii) exchange operation 5 from 1×1 convolution to 3×3 convolution.

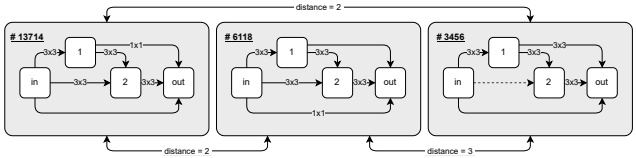

Figure 24. Best architectures in NAS-Bench-201 according to (**left**) clean accuracy, (**middle**) mean adversarial accuracy (over all attacks and ϵ values as described in subsection 2.2), and (**right**) mean common corruption accuracy (over all corruptions and severities) on CIFAR-10. See Figure 1 for cell connectivity and operations.

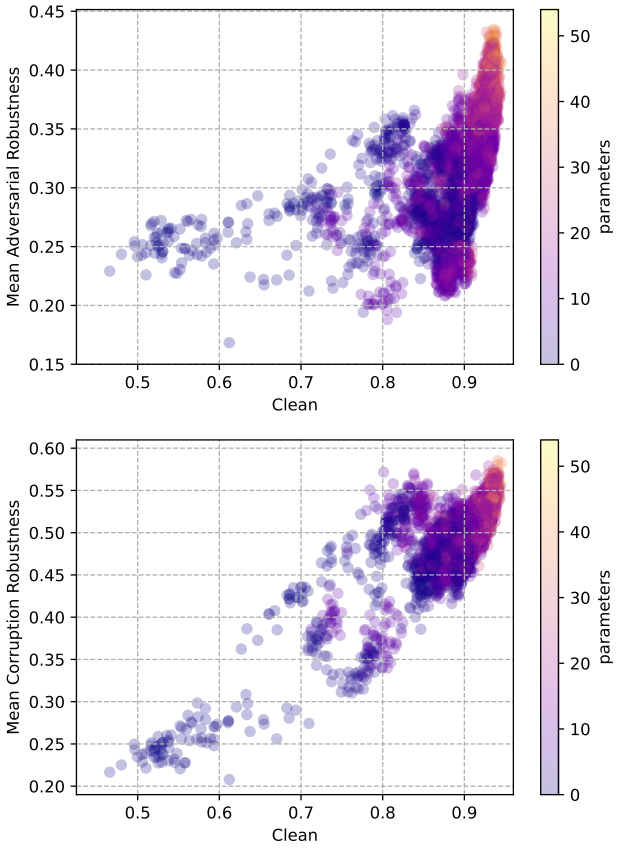

Figure 25. (**left**) Mean adversarial robustness accuracies and (**right**) mean corruption robustness accuracies vs. clean accuracies on CIFAR-10 for all unique architectures in NAS-Bench-201. Scatter points are colored based on the number of kernel parameters of a single cell (1 for each 1×1 convolution, 9 for each 3×3 convolution).

B.2. Cell Kernel Parameter Count

Figure 25 displays the mean adversarial robustness accuracies (left) and the mean corruption robustness accuracies (right) against the clean accuracy, color-coded by the number of cell kernel parameters. We count 1 for each 1×1 con-

volution and 9 for each 3×3 convolution contained in the cell, hence, their number ranges in $[0, 54]$. Since these are multipliers for the parameter count of the whole network, we coin these *cell kernel parameters*. Overall, we can see that the cell kernel parameter count matters in terms of robustness, hence, that networks with large parameter counts are more robust in general. We can also see that the number of cell kernel parameters are more essential for robustness against common corruptions, where the correlation between clean and corruption accuracy is more linear. Also in terms of adversarial robustness, there seems to be a large magnitude of possible improvements that can be gained by optimizing architecture design.

Limited Cell Parameter Count To further investigate the magnitude of possible improvements via architectural design optimization, we look into the scenario of limited cell parameter count.

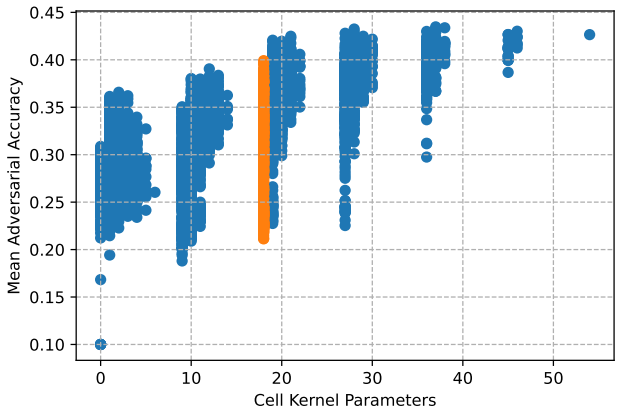

Figure 26. Mean robust accuracy over all attacks as described in subsection 2.2 on CIFAR-10 by kernel parameters $\in [0, 54]$ for all unique architectures in NAS-Bench-201. Orange scatter points depict all architectures with kernel parameter count 18, hence, architectures with exactly 2 times 3×3 convolutions. Although having exactly the same parameter count, the mean adversarial robustness of these networks ranges in $[0.21, 0.40]$.

In Figure 26, we depict all unique architectures in NAS-Bench-201 by their mean adversarial robustness and cell kernel parameter count. Networks with parameter count 18 (408 instances in total) are highlighted in orange. As we can see, there is a large range of mean adversarial accuracies $[0.21, 0.4]$ for the parameter count 18 showing the potential of doubling the robustness of a network with *the same parameter count* by carefully crafting its topology. In Figure 27 we show the top-20 performing architectures (color-coded, one operation for each edge) in the mentioned scenario of a parameter count of 18, according to (**top**) mean adversarial and (**bottom**) mean corruption accuracy. It is interesting to see that in both cases, there are (almost) no convolutions on edges 2 and 4, and additionally no dropping or skipping of edge 1. In the case of edge 4, it seems that a single convolution layer connecting input and output of the cell increases sensitivity of the network. Hence, most of the top-20 robust architectures stack convolutions (via edge 1, followed by either edge 3 or 5), from which we hypothesize that stacking convolutions operations might improve robustness when designing architectures. At the same time, skipping input to output via edge 4 seems not to affect robustness negatively, as long as the input feature map is combined with stacked convolutions. Important to note here is that this is a first observation, which can be made by using our provided dataset. This observation functions as a motivation for how this dataset can be used to analyze robustness in combination with architecture design.

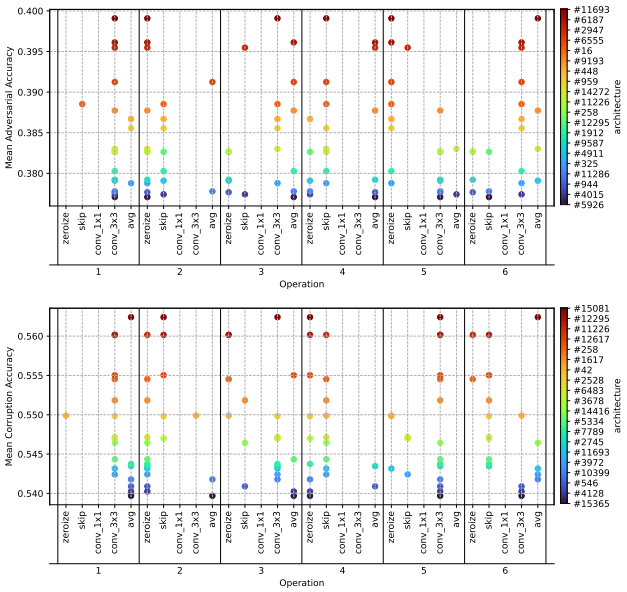

Figure 27. Top-20 architectures with cell kernel parameter count 18 (hence, architectures with exactly 2 times 3×3 convolutions) according to (**top**) mean adversarial accuracy and (**bottom**) mean corruption accuracy on CIFAR-10. See Figure 1 for cell connectivity and operations (1-6).

B.3. Gains and Losses by Single Changes

The fact that our dataset contains evaluations for all unique architectures in NAS-Bench-201 enables us to analyze the effect of small architectural changes. In Figure 28, we depict again all unique architectures by their clean and robust accuracies on CIFAR-10 [10]. The red data point in both plots shows the best performing architecture in terms of clean accuracy (#13714, see Figure 24), while the orange points are its neighboring architectures with edit distance 1. The operation changed for each point is shown in the legend. As we can see in the case of adversarial attacks,

we can trade-off more robust accuracy for less clean accuracy by changing only one operation. While some changes seem obvious (adding more parameters as with 13 and 14), it is interesting to see that exchanging the 3×3 convolution on edge 3 with average pooling (and hence, reducing the amount of parameters) also improves adversarial robustness. In terms of robustness towards common corruptions, each architectural change leads to worse clean and robust accuracy in this case. Changing more than one operation is necessary to improve common corruption accuracy of this network (as we have seen in Figure 24).

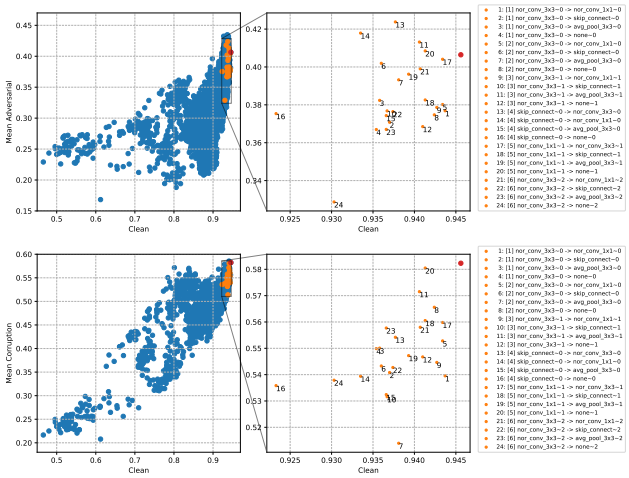

Figure 28. (**top**) Scatter plot clean accuracy vs. mean adversarial accuracy (over all attacks and ϵ values as described in subsection 2.2) on CIFAR-10. (**bottom**) Scatter plot clean accuracy vs. mean common corruption accuracy (over all corruptions and severities) on CIFAR-10. The red data point shows the best performing architecture according to clean accuracy on CIFAR-10. The orange data points are neighboring architectures, where exactly one operation differs. The change of operation is depicted in the legend. The number in brackets refers to the edge where the operation was changed. See Figure 1 for cell connectivity and operations (1-6).