N. Penzel and C. Reimers and P. Bodesheim and J. Denzler:
Investigating Neural Network Training on a Feature Level using Conditional Independence.
ECCV Workshop on Causality in Vision (ECCV-WS). 2022.

Investigating Neural Network Training on a Feature Level using Conditional Independence

Niklas Penzel[1], Christian Reimers[2], Paul Bodesheim[1], Joachim Denzler[1]

[1] Computer Vision Group, Friedrich Schiller University Jena,
Ernst-Abbe-Platz 2, 07743 Jena, Germany,
{niklas.penzel,paul.bodesheim,joachim.denzler}@uni-jena.de
[2] Max Planck Institute for Biogeochemistry, Hans-Knöll-Straße 10,
07745 Jena, Germany, creimers@bgc-jena.mpg.de

Abstract. There are still open questions about how the learned representations of deep models change during the training process. Understanding this process could aid in validating the training. Towards this goal, previous works analyze the training in the mutual information plane. We use a different approach and base our analysis on a method built on Reichenbach's common cause principle. Using this method, we test whether the model utilizes information contained in human-defined features. Given such a set of features, we investigate how the relative feature usage changes throughout the training process. We analyze multiple networks training on different tasks, including melanoma classification as a real-world application. We find that over the training, models concentrate on features containing information relevant to the task. This concentration is a form of representation compression. Crucially, we also find that the selected features can differ between training from-scratch and finetuning a pre-trained network.

Keywords: Training Analysis, Conditional Independence Tests, Explainable-AI, Skin Lesion Classification

1 Introduction

Layering many parameterized functions and non-linearities into deep architectures together with gradient descent, i.e., backpropagation, pushes boundaries in many complex tasks, including classification [13,14,39,40]. However, little is known about how exactly the representations these models learn change during the training process. A better understanding of the model behavior during training could enable us to intervene when wrong features or biases, i.e., spurious correlations, are learned. Additionally, it could benefit the development of new models by identifying and circumventing pitfalls. Hence, understanding the training or, more specifically, how the network dynamics change during training is an important issue [2].

Previous work analyzes the training process by estimating the mutual information between hidden representations and the input or output layer [37,36].

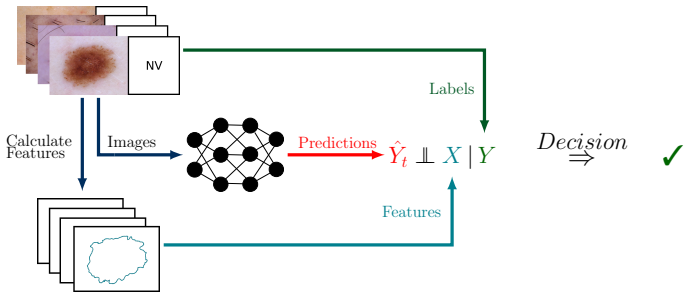

Fig. 1: Overview of our test setup based on the method described in [29]. Here an application for skin lesions is shown. We take a test set, extract human-defined features X, generate predictions $\hat{Y}_t$, and condition both variables on the ground truth Y. The model parameters change over the training steps t possibly leading to a change in predictions $\hat{Y}_t$.

They find that most training is spent on compressing representations to generalize. Shwartz-Ziv and Tishby demonstrated this on smaller toy examples only. We take first steps towards a similar analysis of models training on real-world data, in our case, skin lesion classification. Towards this goal, we base our training analysis on the method described in [29]. This method frames supervised learning as a structural causal model (SCM) after Pearl [24]. Using this SCM and Reichenbach's common cause principle [27], conditional independence tests reveal whether a human-defined feature is relevant for the network prediction. In our work, we rely on three different conditional independence tests and base our analysis on the majority decision to reduce false positives. Figure 1 visualizes our analysis procedure.

This procedure enables us to analyze the training of black-box models without using gradients or hidden representations but directly on a feature level instead. Further, we are not limited to simple input features but can base our analysis on complex features derived as near arbitrary functions of the input.

In this work, we perform an explorative analysis of training three different model architectures on tasks of corresponding complexity. First, we analyze a small multilayer perceptron (MLP) [30] on a small toy example. Second, we investigate a simple convolutional neural network (CNN) [16] learning to distinguish images containing the digits three and eight from the MNIST database [17]. Our final application is the analysis of two EfficientNet-B0's [40] performing melanoma classification on images from the ISIC-archive [1]. Here we make the distinction between training from-scratch and finetuning a pre-trained model.

Overall, we find that our models start with some subset of features and, during training, learn to rely on features containing helpful information for the corresponding task. This behavior can be interpreted as a form of representation compression [37]. We find, however, in our melanoma classification experiments that parameter initialization greatly impacts the final set of selected features. In other words, the model trained from scratch learns to utilize different features

compared to a model pre-trained on ImageNet [32]. Hence, the parameter initialization needs to be further investigated since observing the loss and performance metrics may not be enough to assess the training success.

2 Related Work

In [29], Reimers et al. introduce a method based on conditional dependence to test whether some information contained in a feature is used during the decision process of a trained classifier. They apply their method to MS-COCO [19], CUB200 [43] and HAM10000 [42]. They extend their analysis of skin lesion classifiers in [28] and find that skin lesion classifiers rely on medically relevant data as well as biases. However, Reimers et al. apply their method to fully trained classifiers. To the best of our knowledge, we are the first to apply their method to investigate how the learned representations of a classifier change during training. In the following, we will discuss work related to this objective.

Shwartz-Ziv and Tishby [37] use the information bottleneck theory [41] to analyze the training process of a classifier in the *information plane*. Specifically, the authors estimate the mutual information between the input and a hidden layer and the mutual information between the hidden layer and output. They find that training with SGD splits into two distinct phases: empirical error minimization and representation compression. During the first phase, the observed mutual information between the hidden layer and the output increases, i.e., the network fits the training data. The mutual information between the input and hidden layers decreases in the second phase. Hence, the model focuses on essential features and generalizes. However, they perform their analysis on a small toy example. By contrast, we employ the method of [29] to, among simpler examples, analyze the training process of large models performing skin lesion classification. Hence, we take first steps toward a similar analysis of model training on real-world data. Furthermore, we directly analyze the training on a feature level.

Saxe et al. [34] theoretically analyze the claims by Shwartz-Ziv and Tishby [37] and conclude that the observed split in two phases depends on the used non-linearity, i.e., if the activation function is saturated. They claim that the distinct phases result from the saturated hyperbolic tangent activations that Shwartz-Ziv and Tishby use and would not occur in a network utilizing ReLUs. Additionally, they note that generalization does not necessitate compression. However, Chelombiev et al. [5] develop a more robust adaptive technique to estimate mutual information. They use their method to show that a saturated non-linearity is not required for representation compression. We refer the reader to [36] for more information about the deep information bottleneck theory.

Another approach to analyzing models is to investigate learned concepts, e.g., [15,3]. In [3], Bau et al. propose network dissection, a method to evaluate the alignment of visual concepts with the activations of hidden units in a trained network. They analyze different network architectures and also investigate finetuning between different datasets. They find that over the training, the number of semantic detectors increases, which indicates that the model learns to

recognize more concepts. Bau et al. [3] introduce Broden, a dataset containing annotations for concepts, to perform their analysis. Hence, they rely on some visual interpretation and annotations of what determines a specific concept to be able to test the alignment with hidden units. In contrast, we define features as functions of the inputs alleviating the need for semantic interpretation.

3 Methodology

We argue that the representation compression noted by Shwartz-Ziv and Tishby [37] must also influence the model on a learned feature level. Hence, we propose to analyze the training process by investigating whether the model utilizes information contained in human-defined features. Examples of these features include medically relevant information, e.g., skin lesion asymmetry or border irregularity. Given a set of such features, we expect the model to learn over the training process to discard features that contain no useful information and concentrate on features helpful for solving the task.

The literature describes many methods to test whether a classifier utilizes specific concepts or features, e.g., [15,3,29]. Most of these methods rely on labeled concept datasets and have problems handling features that cannot be visualized in an image, for example, the age of a patient. We choose the method of Reimers et al. [29] as it can handle features that can be defined by near arbitrary functions of the input and gives us the most flexibility when selecting suitable features. It frames supervised learning in a structural causal model [24] and relies on Reichenbach's common cause principle [27] to determine whether the predictions $\hat{Y}$ of a model depend on information contained in a human-defined feature X.

To test whether some information contained in a feature X is used, we test for the conditional dependence of X and $\hat{Y}$ conditioned on the label distribution Y. In practice, we rely on conditional independence (CI) tests to decide if we have to discard the null hypothesis of conditional independence. CI is also intrinsically connected to (conditional) mutual information used by Shwartz-Ziv and Tishby because the (conditional) mutual information is only larger than zero if and only if the variables are (conditionally) dependent on one another.

Figure 1 visualizes the testing procedure for a skin lesion classifier. We refer the reader to [29] for more information and a more detailed introduction.

Note that the method of Reimers et al. [29] cannot detect if a particular feature is causal for a specific model prediction in the sense that changing the feature would lead to a direct change in prediction. However, it can determine if there is any information flow between the feature and the prediction. In other words, if any information contained in the feature is used during inference. Hence, this is a step toward a causal analysis of the training process.

Nevertheless, a critical aspect regarding the performance of the method by Reimers et al. [29] is the choice of CI test. In the following, we will argue why we employ multiple CI tests and form the majority decision.

3.1 Conditional Independence Tests

Many different non-parametric CI tests are described in the literature [18]. These tests are based on various statements equivalent to CI and utilize different properties, e.g., the conditional mutual information [31], or the cross-covariance operator on a reproducing kernel Hilbert space [9]. However, Shah and Peters [35] prove that there is no uniformly valid non-parametric CI test in the continuous case. More specifically, they find that for any such CI test, there are distributions where the test fails, i.e., produces type I errors (false positives).

In our application, we possess little information about the joint distribution and can therefore not rely on domain knowledge to select suitable tests. Hence, we follow the idea described in [28] and rely on a majority decision of a set of CI tests. For our analysis, we chose three tests: Hilbert Schmidt Conditional Independence Criterion (cHSIC) [12,9], Randomized Conditional Correlation Test (RCoT) [38], and Conditional mutual information by k-nearest neighbor estimator (CMIknn) [31]. We selected these tests as they are based on different characterizations of CI. Additionally, we investigated less powerful tests, namely partial correlation and fast conditional independence test [4]. We observed similar behavior. However, both tests are not CI tests in a strict sense. Hence, we omit them from our analysis. In all our experiments we set the significance threshold to $p < 0.01$. To further illustrate our selection of CI tests, we briefly discuss how they work. We also detail our hyperparameter settings for the three selected tests.

cHSIC [9]: HSIC [12] and the conditional version cHSIC [9] are examples of kernel based tests. Note that classical measures such as correlation and partial correlation can only detect linear relationships between variables. Instead of calculating such statistics on the distribution in the original Euclidean space, kernel-based tests utilize the kernel trick [21] to transform the observations into an infinite-dimensional reproducing kernel Hilbert space (RKHS). They then calculate a test statistic in the RKHS, enabling them to capture nonlinear relationships. For HSIC and cHSIC, these test statistics are the Hilbert Schmidt norm of the cross-covariance and the conditional cross-covariance operator, respectively. For more information and definitions of the empirical estimators for the test statistics, we refer the reader to [12,9].

Given the cHSIC test statistic, the actual CI test is a shuffle significance test. First, we calculate the test statistic for the original values of our observed correspondences between features X, predictions $\hat{Y}$ and labels Y. Then we shuffle the values of X and $\hat{Y}$ respectively in separate bins defined by the labels Y and calculate the statistic again. After repeating this process $1,000$ times, i.e., estimating the null distribution, we derive the p-value by comparing the original statistic with the shuffled results. If the p-value is significantly small, i.e., $p < 0.01$, then we have to discard the null hypothesis H_0 and assume conditional dependence instead.

The test statistic is dependent on the selected kernel. In this work we follow Fukumizu et al. [9] and choose a Gaussian radial basis function kernel, i.e.,

$$k_{RBF}(V_1, V_2) = e^{-\frac{1}{2\sigma^2}||V_1 - V_2||^2}, \tag{1}$$

where V_1 and V_2 are two observations of any of our variables. These kernels can differ between our variables X, $\hat{Y}$ and Y depending on the parameter σ. To determine a suitable σ for each of our variables of interest, we use the heuristic proposed by Gretton et al. in [11], i.e.,

$$\sigma_V = \text{median}\{||V_i - V_j||, \forall i \neq j\}, \tag{2}$$

where $V \in \{X, \hat{Y}, Y\}$ is a placeholder for our variables of interest.

RCoT [38]: A known problem of cHSIC is that the null distribution of the test statistic is unknown, and the approximation is computationally expensive [18]. Zhang et al. [46] proposed the kernel conditional independence test (KCIT), an alternative kernel CI test for which the null distribution of the test statistic is derived. KCIT is built with the CI characterization of Daudin [8] in mind, i.e., any residual function contained in certain L^2 spaces of the two test variables X, Y conditioned on the set of conditioning variables Z are uncorrelated. Zhang et al. [46] show that this characterization also holds for functions contained in smaller RKHSs. They use their insight to derive a CI test statistic and the null distribution. Intuitively: KCIT tests whether the correlation of residual functions of the variables conditioned a set of conditioning variables vanishes in an RKHS. Zhang et al. [46] follow Fukumizu et al. [9] and Gretton et al. [11] and use the Gaussian RBF kernel and σ heuristic discussed aboth.

A problem with KCIT is that the computational complexity scales cubically with sample size making it hard to use for larger datasets. However, Strobl et al. [38] propose two approximations of KCIT that perform empirically well and scale linearly. In this work, we use one of these tests, namely RCoT, to approximate KCIT. RCoT utilizes the result of Rahimi and Brecht [26] and approximates the Gaussian RBF kernel used to calculate the test statistic of KCIT with a small set of random Fourier features.

We follow Strobl et al. [38] and use five random Fourier features for our test variables X and $\hat{Y}$ respectively, as well as 25 features for our conditioning variable Y. To select the three kernel widths, i.e., $\sigma_X, \sigma_{\hat{Y}}$ and σ_Y, we again use the heuristic by Gretton et al. [11] (see Equation (2)).

CMIknn [31]: The third test we select for our analysis is CMIknn [31] by Runge. CMIknn is based on yet another characterization of CI: two variables X and Y are conditionally independent given a third conditioning variable Z if and only if the conditional mutual information (CMI) is zero.

Assuming the densities of the variables exist, CMI can be defined using the Shannon entropy H as follows

$$CMI(X, Y; Z) = H_{XZ} + H_{YZ} - H_Z + H_{XYZ}. \tag{3}$$

Runge uses asymptotically unbiased and consistent k-nearest neighbor-based estimators for the necessary entropies to estimate the CMI.

Runge then combines These estimators with a nearest neighbor-based permutation scheme to approximate the distribution of CMI under H_0. A local permutation scheme is necessary to keep the possible dependence between the variables X, Y, and the conditioning variable Z intact.

Both k-nearest neighbor instances introduce a separate hyperparameter k called k_{CMI} and k_{perm} respectively. We follow the settings of Runge [31] for both parameters. To be specific, we set k_{perm} to five and use ten percent of the available samples to estimate CMI, i.e., $k_{CMI} = 0.1 \cdot n$.

4 Selected Classification Tasks and Feature Sets

We investigate three classification tasks of increasing difficulty. This setup enables us to analyze multiple model architectures on tasks of corresponding complexity.

We first propose a simple toy example where we investigate the input features. Second, we train a simple convolutional model on the MNIST [17] images containing either a three or an eight. The corresponding feature set consists of distances to class prototypes. For the third task, we select the real-world problem of skin lesion classification. We analyze medically relevant features and known biases following [28].

Additionally to their complexity, these proposed scenarios differ mainly in how we construct the corresponding feature set. We need a set of features to be *extensive* to gain the most insights. With *extensive* we mean that the combination of features in this set possesses all information necessary to solve the task perfectly, i.e., separate the classes. In other words, an *extensive* set of features contains for each input enough information to correctly classify it and possibly additional information irrelevant to the problem. Given such a set of features and a model able to learn the task, we expect the model to learn which features contain helpful information, i.e., the model extracts useful information from the inputs.

We construct our toy example so that the set of input features meets the above definition of *extensiveness*. However, defining an *extensive* set of features to analyze a high-dimensional real-world task is at least difficult. Hence, for our other proposed tasks, we resort to feature sets containing features we deem helpful to the task and features that should contain little information. This section briefly introduces the three classification tasks and corresponding feature sets in more detail.

4.1 Toy Example

In our first scenario, we want to analyze a small toy example where we can define an *extensive* set of features. Let the input vectors $x_i \in \mathbb{R}^d$ be uniformly sampled from $\mathcal{U}(0,1)^d$. We then split these input vectors into two classes: First,

the positive class contains half of the original sampled set. Second, the negative consists of the vectors not contained in the positive class. Here we randomly select either the first or the last input dimension and flip the value, i.e., multiply by -1.

Figure 1 in Appendix A visualizes the resulting distribution for $d = 2$. In our experiments, we set d to 100 and train a simple multilayer perceptron (MLP) architecture.

Our chosen feature set for this toy example consists of the d input values given by the examples x_i. This feature set is *extensive* in that it contains all information necessary to solve the task because the class only depends on the sign of the first and last input dimension. However, it is not the only possible set that fulfills this requirement Another example would be higher-order polynomial combinations of the input values.

4.2 MNIST

The second scenario is based on the MNIST database [17]. Here we train a simple convolutional network (CNN) to differentiate between images containing a three and an eight. We want to analyze a set of distance features for this simple binary task. Each feature is defined by the cosine distance to a class prototype. The cosine distance for two vectors u and v is defined as

$$1 - \frac{u \cdot v}{||u||_2 \cdot ||v||_2}. \tag{4}$$

We construct this set of features for two reasons. First, distances (4) to class prototypes enable us to analyze the training on arbitrary data.Generating class prototypes enables us to analyze models without domain experts to derive important and unimportant features. Second, the distance to task-relevant class prototypes intuitively contains information helpful in solving the classification task because neural networks learn to separate the input space discriminatively.

We calculate the cosine distance between each image in our test set and the ten MNIST class prototypes. These class prototypes can be seen in Figure 2 in Appendix B. Additionally, we calculate the distance to the ten Fashion-MNIST (F-MNIST) [44] class prototypes to increase the feature variety. Note that to ensure the correct evaluation of the cosine distance, we normalize the class prototypes and our test images with the mean of our test set to shift the center of gravity to the origin.

In total, we have a set of 20 distance features. We expect that over time the model learns to rely only on the distance of an image to the three and eight prototypes. However, the model can use other features as we work under a closed world assumption.

4.3 ISIC-archive

In our third experiment, we analyze models performing a skin lesion classification task following [28]. These are often stated in the form of smaller imbalanced

challenge datasets, e.g., [7,6]. To remedy the problem of few examples, we do not directly train on an available challenge dataset but instead download all images with a corresponding diagnosis from the ISIC-archive [1]. More details and visualizations can be found in Appendix D. In our skin lesion experiments, we simplify the problem further by utilizing only images of the two most common classes: melanomata and benign nevi. Hence, our models train on a binary melanoma detection task with over 33K images.

As a set of features for our analysis, we follow the work of Reimers et al. [28]. We investigate twelve features split into three groups: features containing little helpful information, features containing medically relevant information, and features describing known biases in skin lesion classification. Especially interesting are the features contained in the second group based on the dermatological ABCD-rule [22]. This subset of features contains the medically relevant lesion asymmetry, border roughness, colors, and dermoscopic structures. The bias features include the patients' age, sex, skin color, and the presence of spurious colorful patches in the image. The last subset of features consists of the skin lesion rotation, symmetry regarding a random axis, the id in the ISIC-archive [1], and the MNIST-class corresponding to the skin lesion segmentation mask. For more information on these features or the extraction process, we refer the reader to [28].

For medical tasks, especially skin lesion classification, one often relies on pretraining on large-scale image datasets to compensate for the lack of problem-specific data. However, we want to analyze the training process specifically. Hence, we compare both training from-scratch and using an ImageNet [32] pretrained model.

5 Experiments

In this section, we describe our three architectures learning the classification tasks described in Section 4. After each training setup description, we state our results and note our first insights. We report the proportional usage of features with respect to the corresponding feature set and the gain in performance. Regarding cHSIC and CMIknn, we randomly sample $1,000$ data samples because these two tests scale more than linearly with sample size. Hence, making the number of tests we perform possible.

5.1 MLP Learning a Toy Example

In this first experiment, we train a small MLP on the toy example introduced in Section 4.1. We sample 10K training and 1,000 test examples from the described $d = 100$ dimensional input distribution. The simple MLP model used in this task consists of three hidden layers. The first two layers contain 128 nodes, while the third layer contains 35 nodes. All layers use ReLU [23] activation functions except the output layer, where we employ softmax to generate prediction probabilities. We train the model using SGD with a learning rate of 0.001 and

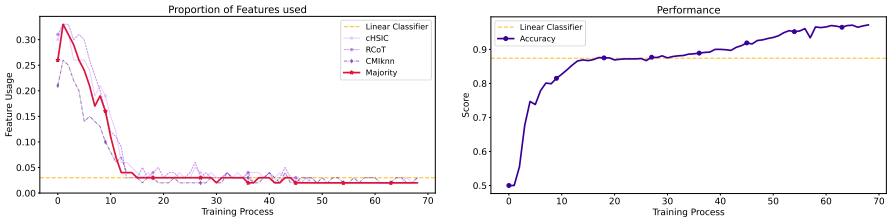

Fig. 2: Visualization of the proportional feature usage throughout the training process. The first plot displays the relative amount of used features compared to a simple linear model. The second plot compares the MLP's accuracy during the training to the accuracy of a linear classifier on the same data.

stop the training early after 67 epochs when we stop observing improvements. Here we estimate the feature usage after every training epoch and report accuracy as the corresponding performance metric. We expect the model to start with some subset of features and learn over time only to utilize the two essential dimensions. Additionally, we compare our MLP to a simple linear model on the same data.

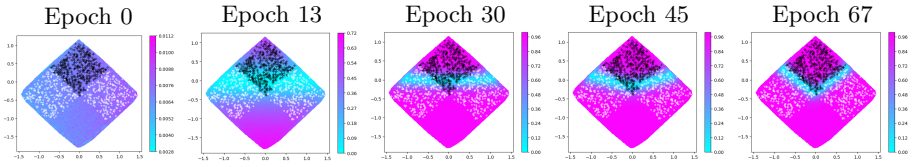

Fig. 3: Visualization of the decision boundary of the MLP during the training on the toy example described in Section 4.1. Black and white points correspond to examples of classes one and two, respectively. Light blue background color marks areas where both classes are equally likely, i.e., the decision boundary. More Information about these plots can be found in Appendix E.

Results: Figure 2 visualizes the relative usage and the improvement in accuracy throughout the training. We observe that the model starts with 0.26 relative usage. Remember that only two features contain helpful information.

The model converges to using three features of our set after some epochs. This moment coincides with the epoch the model reaches the accuracy and feature usage of the linear model. To investigate this further, we visualize the learned decision boundary in Figure 3. We see that the model learns an approximately linear boundary after 13 epochs. At this point, the MLP reaches the linear model accuracy (0.869) and plateaus for around 20 epochs.

During this plateau, we detect the usage of three features similar to the linear model. These are the two essential input features containing task-relevant information and one random feature, likely due to a tilt in the learned boundary. Figure 3 shows that the decision boundary becomes sharper during these 20 plateau epochs of no accuracy improvement.

After around thirty epochs, the model starts to improve notably on the linear baseline. We see that the decision boundary starts to bend. Hence, the MLP focuses more on the two crucial input dimensions. Further, this higher plasticity of the MLP compared to the linear model, i.e., the better fitting boundary, leads to the final feature usage of 0.02 after epoch 45. The MLP correctly identifies the useful inputs and discards useless information leading to improved performance over the linear baseline.

To summarize, the MLP first converges to a linear boundary regarding performance, relative feature usage, and decision boundary. The decision boundary's sharpness increases before the model learns a nonlinear boundary in the next phase. This convergence is also apparent in the relative feature usage, where we can only detect the two critical features after 45 epochs.

5.2 CNN Learning MNIST Three versus Eight

The second task is the binary MNIST three versus eight classification task described in Section 4.2. To learn the task, we rely on a simple CNN consisting of three convolutional layers followed by global average pooling and one fully connected layer. The exact architecture can be found in Table 1 in Appendix C. We use the same hyperparameter settings as in our first experiment but train for 500 epochs. Again we report the accuracy and test the feature usage after every training epoch. We expect that the model prefers the MNIST features over the F-MNIST features and focuses mainly on the three and eight features as they are most relevant to the task.

Results: Figure 4 visualizes the training process and corresponding feature use for our pre-defined distance feature set. Here we see a difference between the three CI tests. CMIknn indicates the usage of nearly all features during the whole training process. However, we expect this to be a failure case as cHSIC and RCoT agree on fewer features [35]. Hence, taking the majority vote leads to fewer false positives.

Regarding relative feature usage, we observe that most features are used in the first few epochs. The distance to the class three prototype is a notable exception. This feature is learned later during the training by the model.

Furthermore, coinciding with an increase in test accuracy, we observe a significant drop in relative feature usage during the training. After 70 epochs, the model only gains marginal improvements in test accuracy. However, on a feature level, we can see that the hidden representation is still changing, and the model focuses on specific features deemed important to the task. This result is similar to our observations in the first experiment (Section 5.1). We observe a similar reduction of feature usage in other binary MNIST tasks.

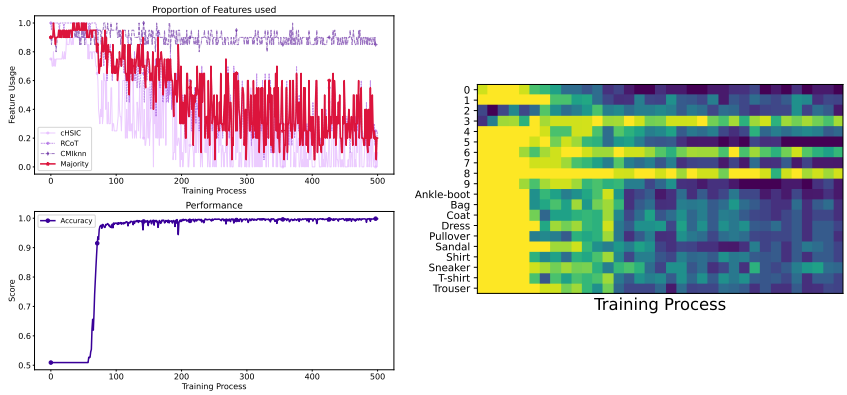

Fig. 4: Usage of the cosine distance features during the training of the MNIST three versus eight binary CNN. The plots on the left-hand side visualize the training, i.e., the relative feature usage (top) and the model accuracy (bottom). The right plot breaks down the detailed use of the 20 cosine distance features. Here a brighter color corresponds to higher usage of the feature.

In the end, the model discards over half of the features but uses approximately equal amounts of MNIST and F-MNIST features. Nevertheless, the detailed feature usage reveals that the model focuses on three distance features during the later stages of training: three, eight, and six. Further, we observe much noise in the feature usage. A possible explanation could be the distribution of the observed cosine distances. Figure 4 in Appendix F shows that the average cosine distances of our test set to the class prototypes are very similar for all prototypes except the three and the eight. Hence, the observed noise in our visualizations could be due to the small signal-to-noise ratio between the different features.

However, the usage of the six feature is unexpected and needs to be further investigated. In the first experiment, we saw that the internal representation can still change even if we do not immediately detect a change in performance. Hence, a possible explanation could be that we stopped training too early.

5.3 Modern Architecture Learning Melanoma Classification

To apply our analysis to a complicated real-world example, we selected the skin lesion classification task described in Section 4.3. For our analysis, we choose an EfficientNet-B0 architecture [40]. Appendix G details our model selection process. To additionally improve the balanced accuracy, we use the loss imbalance weights presented in [45] with $\alpha = 1.2$. We also employ the learning optimal sample weights (LOW) mechanism [33] on top of the standard categorical cross-entropy loss to boost our model performance.

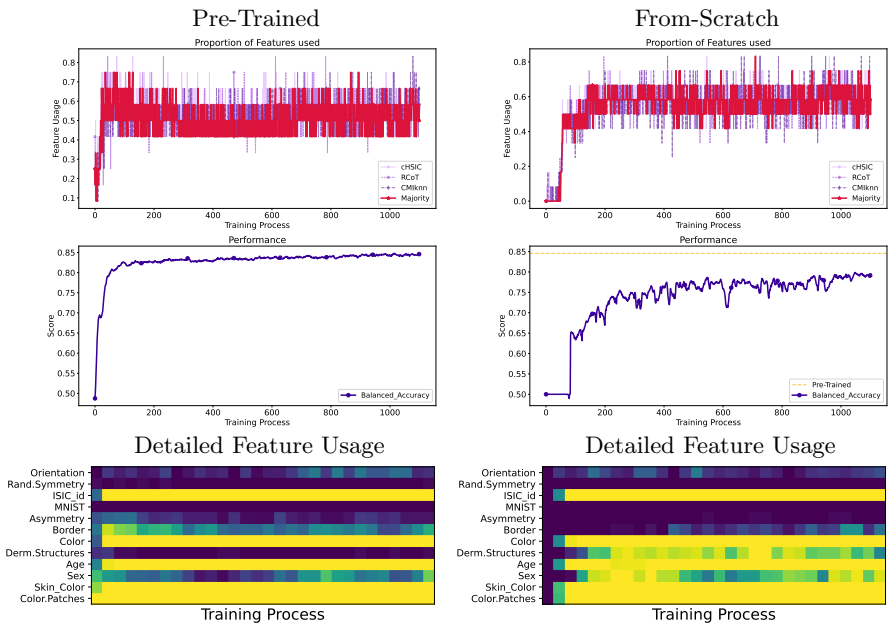

Fig. 5: The plots on the left-hand side visualize the training process of the ImageNet [32] pre-trained EfficientNet-B0, while the right column visualizes the equivalent for the model trained from-scratch. The rows are organized in the following way: First, the relative feature usage, second, the change in balanced accuracy, and third detailed feature usage. The x-axis in all visualizations corresponds to the training process. In the detailed feature usage visualizations, a brighter color denotes higher feature usage.

We use a batch-wise time resolution for our training process analysis. In other words, after every training batch[3], we evaluate the usage of the twelve features discussed in Section 4.3. We report the balanced accuracy, i.e., average class-wise accuracy, as a performance metric in this imbalanced setting.

Results: Figure 5 visualizes the training process and the feature usage. The pre-trained model already uses some (25%) features after being initialized with the pre-trained weights. In contrast, the from-scratch trained model starts without knowing how to extract any information contained in our human-defined feature set. This observation is not surprising given the high complexity of our features as well as the random initialization. Some features likely contain information helpful for classification in general, so it is not surprising that the pre-trained model already shows some response for the colorful patches feature.

After approximately 200 batches, both models converge to ≈ 0.6 relative feature usage. Given our limited feature set that is in no way extensive, we see an

[3] Time per update step: ≈ 0.24s, time per CI test: ≈ 2s.

increase in feature usage. This observation is contrary to our previous experiments (Section 5.1, Section 5.2), where we analyzed simpler features. However, our skin lesion feature sets includes more complex high level features that are very task specific, e.g., dermoscopic structure occurrence. Hence, it is not surprising that the networks need some time to learn representations that capture information contained these features. This observation is similar to the increase in concept detectors noted by Bau et al. in [3].

Generally, the pre-trained model training is more stable, and we do not observe large fluctuations between single batches. It also converges towards a higher balanced accuracy. This observation is well known and likely due to the better initialization of the pre-trained model.

Let us now analyze the detailed feature usage. We find that both models use the ISIC id feature that should contain no information. However, this observation stems most likely from the limitation the authors noted in [28]. The ISIC id is a proxy feature for the dataset from which the skin lesion image originated because the ids are consecutive numbers and we use the complete ISIC-archive.

Further, as we expected, both models learned to utilize the four bias features, especially during the last training batches, similar to the results in [28]. However, we find interesting behavior in the subset of medically relevant features. Here both models heavily rely on the color feature. The pre-trained model seems to rely more on the border irregularity of the skin lesions. In contrast, the model trained from-scratch learns to utilize the presence of dermoscopic structures. To ensure the validity of this statement, we investigated a second initialization of the model trained from-scratch. The results of the second initialization are very similar and can be found in Appendix H. To summarize, we find that pre-training influences which features a model will utilize during inference. Hence, the initialization seems to impact the learned representations greatly, even on a feature level. In fact pre-training seems to prevent the model from learning an expert annotated feature deemed useful for detecting melanomata by dermatologists [22].

6 Conclusions

We employ the method of Reimers et al. [28] to analyze feature usage of varying feature sets in our three tasks. This approach enables us to analyze the training of three different architectures. The models in our first two scenarios (Section 5.1, Section 5.2) start with more features than are helpful. During the training process, the models converge to a set of features containing information useful for the task. This convergence towards a small subset of features can be interpreted as representation compression after Shwartz-Ziv and Tishby [37].

The advantage of our methodology is that we can apply it to analyze more complex real-world scenarios. During our analysis of melanoma classification (Section 5.3), we find that both EfficientNets increase the number of features they utilize over the training. This observation is likely due to the higher complexity

of the corresponding feature set. Our results are comparable to the increase in concept detectors over the training process noted by Bau et al. [3].

Additionally, we find that the network initialization has a considerable impact. Even though the model trained from-scratch and the pre-trained model converge on a similar proportional usage of human-defined features, both differ in the medically relevant feature subset. This observation opens the question if we can enforce the usage of certain features and use the knowledge of domain experts. Another direction for future research is the construction of extensive feature sets for analysis purposes.

References

1. International skin imaging collaboration, ISIC Archive. https://www.isic-archive.com/, https://www.isic-archive.com/
2. Alain, G., Bengio, Y.: Understanding intermediate layers using linear classifier probes. arXiv preprint arXiv:1610.01644 (2016)
3. Bau, D., Zhou, B., Khosla, A., Oliva, A., Torralba, A.: Network dissection: Quantifying interpretability of deep visual representations. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 6541–6549 (2017)
4. Chalupka, K., Perona, P., Eberhardt, F.: Fast conditional independence test for vector variables with large sample sizes. arXiv preprint arXiv:1804.02747 (2018)
5. Chelombiev, I., Houghton, C., O'Donnell, C.: Adaptive estimators show information compression in deep neural networks. arXiv preprint arXiv:1902.09037 (2019)
6. Codella, N., Rotemberg, V., Tschandl, P., Celebi, M.E., Dusza, S., Gutman, D., Helba, B., Kalloo, A., Liopyris, K., Marchetti, M., Kittler, H., Halpern, A.: Skin Lesion Analysis Toward Melanoma Detection 2018: A Challenge Hosted by the International Skin Imaging Collaboration (ISIC). arXiv:1902.03368 [cs] (Mar 2019), http://arxiv.org/abs/1902.03368, arXiv: 1902.03368
7. Codella, N.C., Gutman, D., Celebi, M.E., Helba, B., Marchetti, M.A., Dusza, S.W., Kalloo, A., Liopyris, K., Mishra, N., Kittler, H., et al.: Skin lesion analysis toward melanoma detection: A challenge at the 2017 international symposium on biomedical imaging (isbi), hosted by the international skin imaging collaboration (isic). In: 2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018). pp. 168–172. IEEE (2018)
8. Daudin, J.: Partial association measures and an application to qualitative regression. Biometrika **67**(3), 581–590 (1980)
9. Fukumizu, K., Gretton, A., Sun, X., Schölkopf, B.: Kernel measures of conditional dependence. Advances in neural information processing systems **20** (2007)
10. Gessert, N., Nielsen, M., Shaikh, M., Werner, R., Schlaefer, A.: Skin lesion classification using ensembles of multi-resolution efficientnets with meta data. MethodsX **7**, 100864 (2020)
11. Gretton, A., Borgwardt, K., Rasch, M., Schölkopf, B., Smola, A.: A kernel method for the two-sample-problem. Advances in neural information processing systems **19** (2006)
12. Gretton, A., Fukumizu, K., Teo, C.H., Song, L., Schölkopf, B., Smola, A.J., et al.: A kernel statistical test of independence. In: Nips. vol. 20, pp. 585–592. Citeseer (2007)
13. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
14. 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)
15. Kim, B., Wattenberg, M., Gilmer, J., Cai, C., Wexler, J., Viegas, F., et al.: Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In: International conference on machine learning. pp. 2668–2677. PMLR (2018)
16. LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Backpropagation applied to handwritten zip code recognition. Neural Computation **1**(4), 541–551 (1989). https://doi.org/10.1162/neco.1989.1.4.541

17. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)
18. Li, C., Fan, X.: On nonparametric conditional independence tests for continuous variables. Wiley Interdisciplinary Reviews: Computational Statistics **12**(3), e1489 (2020)
19. Lin, T.Y., Maire, M., Belongie, S.J., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: ECCV (2014)
20. van der Maaten, L., Hinton, G.: Visualizing data using t-sne. Journal of Machine Learning Research **9**(86), 2579–2605 (2008), `http://jmlr.org/papers/v9/vandermaaten08a.html`
21. Mercer, J.: Functions of positive and negative type and their connection with the theory of integral equations. Philos. Trans. Roy. Soc. London **209**, 415–446 (1909)
22. Nachbar, F., Stolz, W., Merkle, T., Cognetta, A.B., Vogt, T., Landthaler, M., Bilek, P., Braun-Falco, O., Plewig, G.: The ABCD rule of dermatoscopy. High prospective value in the diagnosis of doubtful melanocytic skin lesions. Journal of the American Academy of Dermatology **30**(4), 551–559 (Apr 1994). https://doi.org/10.1016/s0190-9622(94)70061-3
23. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th International Conference on International Conference on Machine Learning. p. 807–814. ICML'10, Omnipress, Madison, WI, USA (2010)
24. Pearl, J.: Causality. Cambridge university press (2009)
25. Perez, F., Vasconcelos, C., Avila, S., Valle, E.: Data augmentation for skin lesion analysis. In: OR 2.0 Context-Aware Operating Theaters, Computer Assisted Robotic Endoscopy, Clinical Image-Based Procedures, and Skin Image Analysis, pp. 303–311. Springer (2018)
26. Rahimi, A., Recht, B.: Random features for large-scale kernel machines. In: NIPS (2007)
27. Reichenbach, H.: The direction of time. University of California Press (1956)
28. Reimers, C., Penzel, N., Bodesheim, P., Runge, J., Denzler, J.: Conditional dependence tests reveal the usage of abcd rule features and bias variables in automatic skin lesion classification. In: CVPR ISIC Skin Image Analysis Workshop (CVPR-WS). pp. 1810–1819 (June 2021)
29. Reimers, C., Runge, J., Denzler, J.: Determining the relevance of features for deep neural networks. In: European Conference on Computer Vision. pp. 330–346. Springer (2020)
30. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. nature **323**(6088), 533–536 (1986)
31. Runge, J.: Conditional independence testing based on a nearest-neighbor estimator of conditional mutual information. In: International Conference on Artificial Intelligence and Statistics. pp. 938–947. PMLR (2018)
32. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. International journal of computer vision **115**(3), 211–252 (2015)
33. Santiago, C., Barata, C., Sasdelli, M., Carneiro, G., Nascimento, J.C.: Low: Training deep neural networks by learning optimal sample weights. Pattern Recognit. **110**, 107585 (2021)
34. Saxe, A.M., Bansal, Y., Dapello, J., Advani, M., Kolchinsky, A., Tracey, B.D., Cox, D.D.: On the information bottleneck theory of deep learning. Journal of Statistical Mechanics: Theory and Experiment **2019**(12), 124020 (2019)

35. Shah, R.D., Peters, J.: The hardness of conditional independence testing and the generalised covariance measure. The Annals of Statistics **48**(3), 1514–1538 (2020)
36. Shwartz-Ziv, R.: Information flow in deep neural networks. arXiv preprint arXiv:2202.06749 (2022)
37. Shwartz-Ziv, R., Tishby, N.: Opening the black box of deep neural networks via information. arXiv preprint arXiv:1703.00810 (2017)
38. Strobl, E.V., Zhang, K., Visweswaran, S.: Approximate kernel-based conditional independence tests for fast non-parametric causal discovery. Journal of Causal Inference **7**(1), 20180017 (2019). https://doi.org/doi:10.1515/jci-2018-0017, `https://doi.org/10.1515/jci-2018-0017`
39. 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)
40. Tan, M., Le, Q.: Efficientnet: Rethinking model scaling for convolutional neural networks. In: International conference on machine learning. pp. 6105–6114. PMLR (2019)
41. Tishby, N., Pereira, F.C., Bialek, W.: The information bottleneck method. ArXiv **physics/0004057** (2000)
42. Tschandl, P., Rosendahl, C., Kittler, H.: The ham10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. Scientific data **5**(1), 1–9 (2018)
43. Welinder, P., Branson, S., Mita, T., Wah, C., Schroff, F., Belongie, S.J., Perona, P.: Caltech-ucsd birds 200 (2010)
44. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747 (2017)
45. Yao, P., Shen, S., Xu, M., Liu, P., Zhang, F., Xing, J., Shao, P., Kaffenberger, B., Xu, R.X.: Single model deep learning on imbalanced small datasets for skin lesion classification. IEEE Transactions on Medical Imaging (2021)
46. Zhang, K., Peters, J., Janzing, D., Schölkopf, B.: Kernel-based conditional independence test and application in causal discovery. arXiv preprint arXiv:1202.3775 (2012)

Investigating Neural Network Training on a Feature Level using Conditional Independence - Appendix -

Niklas Penzel[1], Christian Reimers[2], Paul Bodesheim[1], Joachim Denzler[1]

[1] Computer Vision Group, Friedrich Schiller University Jena,
Ernst-Abbe-Platz 2, 07743 Jena, Germany,
{niklas.penzel,paul.bodesheim,joachim.denzler}@uni-jena.de
[2] Max Planck Institute for Biogeochemistry, Hans-Knöll-Straße 10,
07745 Jena, Germany, creimers@bgc-jena.mpg.de

A Toy example visualization

Figure 1 visualizes a two dimensional version of the toy example described in Section 4.1.

B MNIST and F-MNIST Feature Set

Figure 2 displays the class prototypes of the MNIST [17] database and the Fashion-MNIST (F-MNIST) [44] dataset. Additionally, the normalized prototypes are shown after we center both datasets by subtracting the mean of all images in our test set. The distances to these normalized prototypes are the features we use for our analysis in Section 5.2.

C MNIST 3 versus 8 CNN

The architecture we employ in our analysis of the MNIST [17] three versus eight tasks is a simple CNN architecture listed in Table 1. All layers use ReLU activation functions except the output layer, where we use softmax to generate prediction probabilities.

Table 1: The architecture of the simple CNN we use for our MNIST experiments.

Layertype	Filter Size	Filters	Padding
Convolutional	5×5	8	2×2
Convolutional	3×3	16	1×1
Convolutional	3×3	16	1×1
GA Pooling	-	-	-
Fully-Connected	1×1	128	-

D ISIC-archive Details

In this section, we list some additional details about the ISIC-archive[3]. The ISIC-archive contains 69,445 skin lesion images of various datasets and research groups. For 41,941 of these images, diagnosis data is available. Table 2 list different diagnoses counts. A t-SNE [20] visualization of the ISIC-archive can be found in Figure 3. In this work, we focus our analysis on the over 33K melanoma and nevus images. We construct a binary classification task between these two classes and split the dataset into training, validation, and test data using the ratios 0.7, 0.1, and 0.2, respectively.

Table 2: The number of images for varying diagnoses in the ISIC-archive.

Diagnosis	**Count**
scar	1
cafe-au-lait macule	1
angiofibroma or fibrous papule	1
other	10
atypical melanocytic proliferation	14
angioma	15
lentigo simplex	27
lichenoid keratosis	32
lentigo NOS	111
dermatofibroma	246
vascular lesion	253
solar lentigo	270
squamous cell carcinoma	656
actinic keratosis	869
pigmented benign keratosis	1,099
seborrheic keratosis	1,464
basal cell carcinoma	3,396
melanoma	5,598
nevus	27,878

E Decision Boundary Visualization

Figure 3 displays the progression of the learned decision boundary by the MLP. We utilize PCA to reduce the dimension of our 100-dimensional toy example down to two dimensions to make a visualization possible. This approach is possible as we know that the two axes with the highest variance perfectly encapsulate our data because the latent distribution consists of three noise cubes (compare Figure 1 in Appendix A).

[3] `https://www.isic-archive.com/`

We utilize our knowledge of the latent distribution to generate the decision boundary visualizations displayed in Figure 3. We sample 200K points of the problem distribution and generate predictions with the current MLP. Let l_+ and l_- be the activations of the logits corresponding to the two classes. Then we calculate for all predictions the squared logit difference as $(l_+ - l_-)^2$. Afterward, we use the PCA projection matrix learned on our original test data to project these 200K points into our two-dimensional visualization space. The squared logit difference with the projected points can be interpreted as a height map, where zero encodes the decision boundary of the MLP. We visualize this height map using linear triangulation as the background color in Figure 3.

F Average Cosine Distance

Figure 4 displays the average cosine distances of the MNIST three versus eight test images to the 20 class prototypes. We can see two outliers, the three and the eight prototypes, which are, on average closest to the respective class and farthest to the opposite class. However, the other images show similar distances and cannot be easily distinguished. Further, we do not observe a notable difference in the average distance between the MNIST and the F-MNIST prototypes. This fact could be a possible explanation for the observed noise in Section 5.2.

G Skin Lesion Classification Model Selection

To select a suitable model for skin lesion classification, we investigate two different architectures: ResNet50 [13] and EfficientNet-B0 [40]. We chose a ResNet model because it is the best performing architecture for melanoma classification in [25], out of ResNet [13], InceptionNet-v4 [39], and DenseNet [14]. Similarly, we selected an EfficientNet model following [10]. Gessert et al. [10] build large ensembles out of various model architectures and find that EfficientNets consistently perform well for skin lesion classification.

To select an architecture for our analysis, we trained the two described models for 100 epochs on our described melanoma versus nevus task.

Table 3: Accuracy and balanced accuracy for two widely used architectures. We compare from-scratch training and finetuning a pre-trained model. Additionally, we report the standard deviation.

	ResNet50 [13]		**EfficientNet-B0** [40]	
Paradigm	ACC	bACC	ACC	bACC
Pre-Trained	0.886 ± 0.007	0.736 ± 0.031	0.924 ± 0.003	0.829 ± 0.013
From-Scratch	0.883 ± 0.003	0.713 ± 0.015	0.905 ± 0.009	0.785 ± 0.024

The results can be found in Table 3. We find that the EfficientNet architecture significantly outperforms the ResNet for both from-scratch training and finetuning a pre-trained model. Additionally, the EfficientNet is faster to train due to having less than half the number of parameters. Hence, we select an EfficientNet-B0 for our training analysis.

H Second From-Scratch Training Initialization

Figure 5 displays the training visualizations for a second randomly initialized EfficientNet-B0 trained from-scratch. The learned features as well as the change in proportional usage is remarkably similar compared to the right coloumn in Figure 5. Both initializations lead to very different feature usage when compared to the pre-trained model. Additionally, here the model also learns to incorporate the lesion orientation, which could hint at an additional dataset bias. Hence, further investigation is necessary.

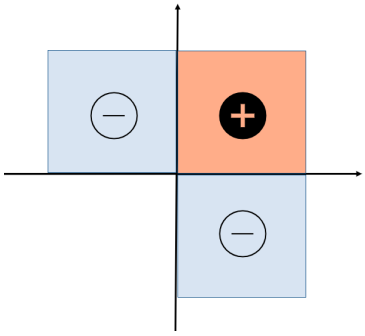

Fig. 1: Visualization of our proposed toy example. Here the problem dimension d is 2. Note that this dimension implies that both input features are essential to solve the problem.

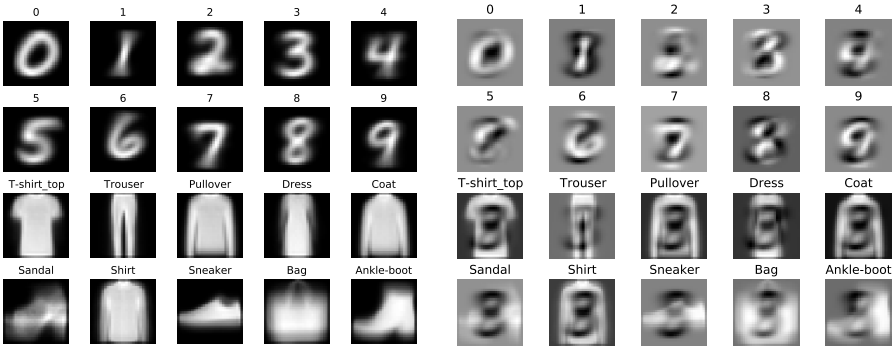

Fig. 2: MNIST [17] and F-MNIST [44] class prototypes. The right-hand prototypes are normalized by the test images of the MNIST database displaying either a three or an eight. This step is necessary to ensure that cosine distance can be calculated correctly.

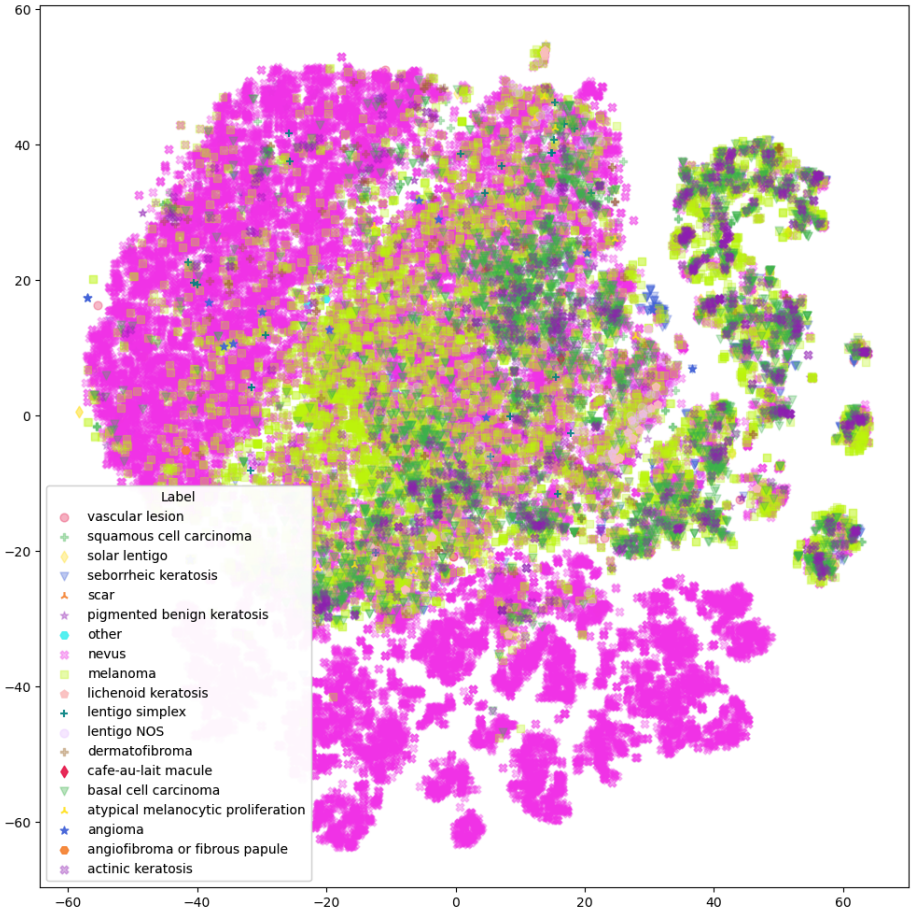

Fig. 3: A t-SNE [20] visualization of all images in the ISIC-archive that have available diagnosis data. The majority of images are either *melanomata* or *nevi*.

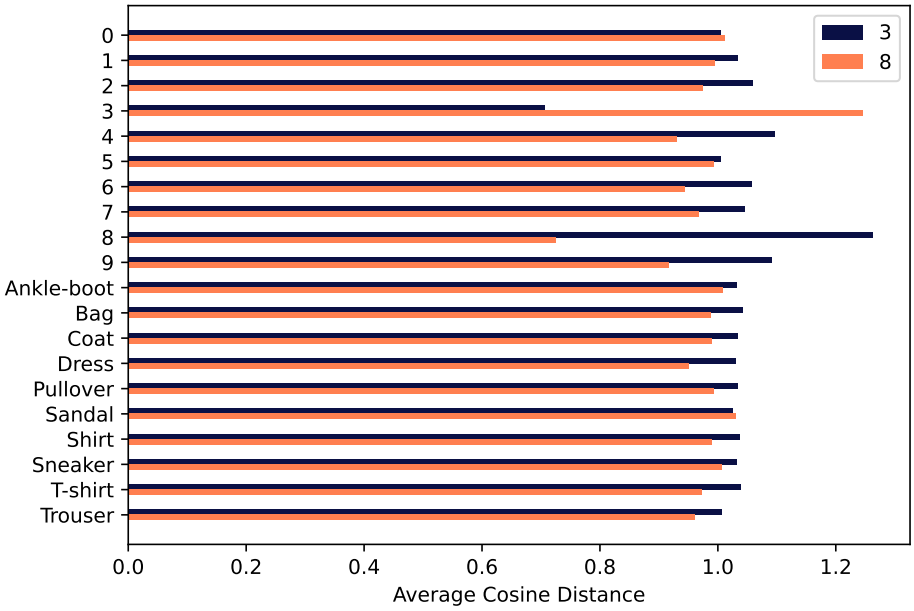

Fig. 4: Average cosine distance of the MNIST three versus eight test images to the twelve class prototypes. The test images are split into two groups according to the corresponding class label.

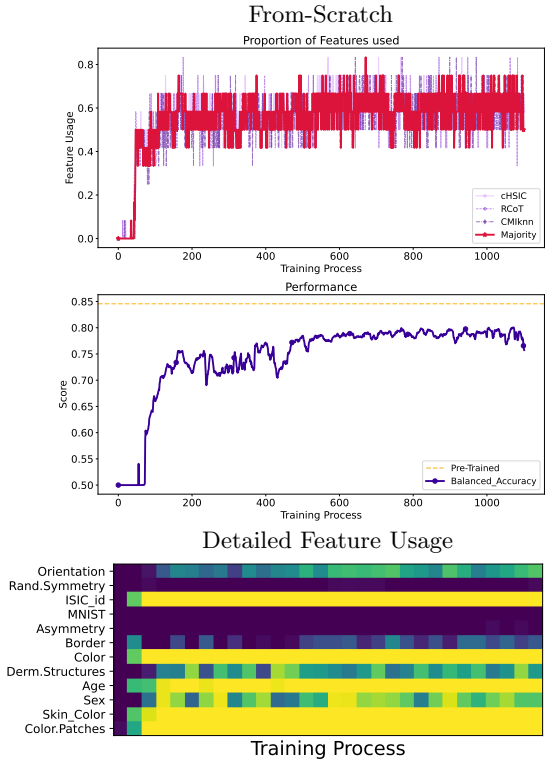

Fig. 5: Visualization of a second random initialization of the EfficientNet-B0 trained from-scratch. First, the relative feature usage is shown. Second, the change in balanced accuracy, and third detailed feature usage is visualized. The x-axis in all visualizations corresponds to the training process. In the detailed feature usage visualizations, a brighter color denotes higher feature usage.