2D Image Classification of Steel Strip Surface Defects

Zorigto Dorzhiev, Florian Feiler

***Abstract*—The NEU surface defect dataset is a standard 2D image dataset for image classification and object detection. Multiple deep-learning based approaches were applied in the past to correctly classify the six types of surface defects, exceeding each other in complexity.**
We are comparing the performance of a widely used convolutional neural network (CNN), the VGG16, with a simple custom CNN and a simpler variant of the VGG16.
Our comparison suggests that the complexity of the VGG16 and similar CNNs is not necessary for achieving high accuracy on such a dataset, only creating computational overhead and wasting resources.

***Index Terms*—surface defect, defect classification, CNN, multi-label classification.**

I. Introduction

HOT-ROLLED steel strip is a metal product that is widely used in various industries, ranging from heavy industry and shipbuilding to construction and metal forging.

A. Steel Defects

The quality of hot-rolled steel strip is an important factor that can affect all of the following processes. Manufacturing technologies are not perfect that is why different types of defects on the steel strip can occur during the production process. These defects lower the quality and reduce shelf life of steel products. Defect detection on early stages is vital to increase overall performance. Modern techniques to solve this problem should be applied because manual inspection of hot-rolled steel strips takes a lot of time and has high rate of errors. These restrains require just-in-time detection, which is both reliable and accurate.

B. NEU Surface Defect Dataset

The Northeastern University (NEU) provides a database of surface defects of hot-rolled steel strips [1]. There are six types of defects in this dataset such as crazing (Cr), inclusion (In), patches (Pa), pitted surface (PS), rolled-in scale (RS) and scratches (Sc). The database includes 300 samples for each type of defect, totally resulting in 1800 grayscale images. Different types of defects are shown in Fig. 1. Each image's resolution 200 × 200 px.

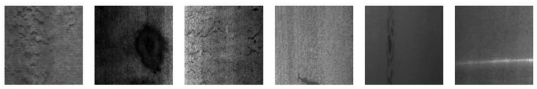

Figure 1: Types of defects on the surface of the hot-rolled steel strip. From left to right: rolled-in scale, patches, crazing, pitted surface, inclusion, scratches.

As the authors mention, the dataset exhibits two difficult challenges. Firstly, there are large variations in appearance within a class (high intra-class variation), while different classes have similar aspects (high-inter class similarity), as shown in Fig. 2. Secondly, the NEU surface defect database is rather small dataset that suffer from the influence of illumination and material changes.

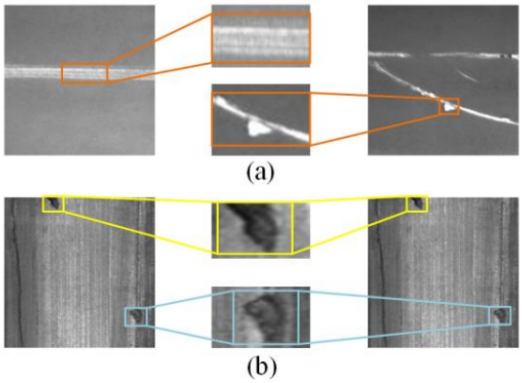

Figure 2: NEU surface defect database challenges. (a) High intra-class variation. (b) High inter-class similarity. [2]

C. Contribution

Zorigto Dorzhiev: Introduction, Related Works, Multi-Label Classification
Florian Feiler: CNN Methods, Custom CNN Architecture, VGGs, Single-Label Classification, Conclusion

Report submitted May 19, 2020

II. Related Works

A. A noise robust method based on completed local binary patterns for hot-rolled steel strip surface defects, 2013

First attempts to surface defect recognition were carried out in 2013 by K. Song and Y. Yang [1]. They proposed an automated surface inspection system based on local binary patterns (LBP) which is used for texture classification. The feature extraction subsystem presented the adjacent evaluation completed local binary patterns (AECLBPs). This method was introduced in order to overcome feature instability and the noise sensitivity. To classify the extracted features a variety of classifiers were used such as SVM and nearest neighbor classifier. The NEU surface defect database was constructed to test the proposed AECLBP approach and compare with the basic LBP methods. The experimental results showed that the new approach improved the recognition rate and achieved 98.87% accuracy. Additionally, the AECLBP method is more robust to additive Gaussian noise.

B. Research on Classification of Surface Defects of Hot-rolled Steel Strip Based on Deep Learning, 2019

C. Wang *et al.* proposed a deep learning model for classification of surface defects [3]. The core of the model is a simple feed-forward neural network that consists of convolutional layers, pooling layers and fully convolutional layers. They achieved an accuracy of $98.6\,\%$ on the NEU surface defect database with a recognition speed of $60\,\text{ms}$ per defect. The authors consider a practical application prospect of their model which can meet the demand of real-time detection in industrial production.

C. PGA-Net: Pyramid Feature Fusion and Global Context Attention Network for Automated Surface Defect Detection, 2019

H. Dong *et al.* introduce a pyramid feature fusion module based on convolution with different kernel sizes (PGA-Net), which fuses multi-level features from all stages of backbone CNN into multi-scale resolutions [2]. They designed a global context attention module to guarantee efficient information transfer from low-resolution to high-resolution. The surface defect detection task is considered a pixel-wise task in this paper. The PGA-net method achieves mean intersection-over-union (mIoU) of 82.15% on NEU-Seg Dataset with the detection speed of $41 - 49\,\text{fps}$.

D. An End-to-end Steel Surface Defect Detection Approach via Fusing Multiple Hierarchical Features, 2020

Aside from defect classification there is the task to find the exact location for each defect in an image. Combining two tasks is still challenging in practice. Yu He *et al.* use deep network ResNet as the backbone in their system [4]. Pretraining large networks on huge datasets, such as ImageNet, is the key of this work to reach the better performance in the classification task. To avoid the loss of details in the last convolutional feature map the presented method fuses multiple feature maps at each stage of the CNN. The established Defect Detection Network (DDN) show the class scores and the bounding-box coordinates. DDN method with ResNet50 achieves the highest mean average precision of 82.3% for defect detection. For classification, the best result is 99.67% with ResNet and multilevel-feature fusion network (MFN).

III. Convolutional Neural Networks

For classification, machine learning models need to have good representations of the data. For using traditional models, representations (features) need to be handcrafted by feature extraction and engineering. For the NEU surface dataset, such features could be gray levels distribution, contrast, homogeneity, or asymmetry. Various projects using such approaches yield small accuracy up to $80\,\%$ [5]. Therefore, such simple models were not able to form meaningful representations of the data. CNNs enable learning representations instead of crafting them, which generally lead to more accurate classification results.
Therefore, we approached the classification problem in two ways. Firstly, we trained a custom CNN to understand which accuracy a very simple model could achieve. Secondly, we used transfer learning for obtaining a robust network that is faster to train. This section will provide an overview over the methods used for both approaches and will introduce and motivate the chosen architectures. In the end we will specify the architecture used for multi-label classification.

A. Methods

To deal with the previously mentioned shortcomings of the dataset, some non-trivial methods need to be used to enable training a deep neural network. Particularly, we chose three methods for training a small CNN, which simultaneously serve both as regularization methods and allow us to effectively train the network. Furthermore, transfer learning is introduced.

1) Data Augmentation: One of the downsides of using Deep Learning methods is that models need to be trained on extensive datasets to generalize well. When dealing with relatively small real world datasets like NEU surface defects, networks will be likely to overfit on the training data: for example if the camera is always aligned in the same angle and all pictures are taken in the same illumination setting, the network would fail to classify different defect types in another setting with different illumination and alignment. To prevent such an overfitting, it is useful to use data augmentation. There are multiple variants of transforming images, among which are color jittering, object translation, blending, random rotation, cropping, and zooming. For our purposes, we have chosen to rotate input images randomly by a multiplicity of $90°$. Thereby, we use an augmented dataset of four times its original size.

2) BatchNorm: Batch normalization (BatchNorm) describes the process of normalizing each batch's data to a normal distribution $\mathcal{N}(\mu = 0, \sigma^2 = 1)$ for each layer. BatchNorm as a regularization method was originally introduced to reduce a quantity called internal covariate shift [6], which describes the shift of activation distributions in each layer through the network. The motivation for using BatchNorm was to whiten these distributions per batch to reduce this shift and thereby allow the network to converge faster.

Even though current research shows that reducing internal covariate shift practically does not hinder convergence of a network [7], BatchNorm allows networks to be less sensitive to hyper-parameters, converge faster, and learn on higher learning rates. It is suggested that these effects occur due to a smoothing of the optimization landscape, which expresses as a decreased variation of the loss value and an enhanced gradient predictiveness.

BatchNorm is useful during training, during testing it can be completely absorbed into the network's weights. Therefore, we use this method only during training the custom CNN.

3) Dropout: Ensemble models allow different parts of the model to specialize on different aspects of data analysis.

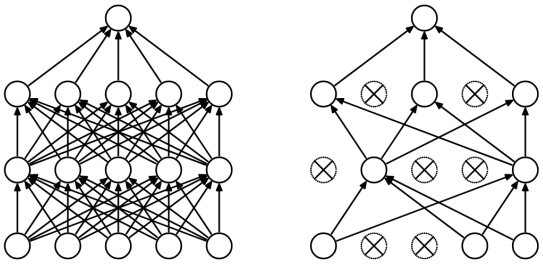

Figure 3: Neural network with dropout during inference and training time. During training, a fraction of all nodes get randomly chosen and deactivated. Thereby, an ensemble of models is trained. At inference time, model averaging is performed by activating all nodes. Taken from [8].

One way to achieve this is to randomly deactivate a fraction of nodes in a network during training time. Thereby, in every training step a new sub-network learns from training data. During testing all nodes are activated and as a result, the network achieves a lower generalization error [9]. It is important to adjust input during inference to keep the average activation level constant.

4) Transfer Learning: The better the representation of input data the better the model is able to generalize. Small datasets such as NEU surface dataset lack of variety and repetition to train a very accurate CNN from scratch. Models are not able to learn useful high-level representations on such small datasets. One approach at overcoming this problem is to use transfer learning. If there is a bigger datasets with similar statistics as the small dataset, it is possible to learn meaningful representations on the bigger dataset and then transfer the model to the smaller dataset. Thereby, the model is able to quickly generalize even from the small dataset by only fine-tuning the previously learned representations. Our networks use pre-trained models on ImageNet, which was trained on 1.2 million images with 1000 classes of animals. Due to the difference of surface defects and animals, we chose to not only retrain the output layer, but also to fine-tune the entire model. Thereby, we allow the network to adapt to the new input types, but still use the high-level representations of the pre-training.

B. Custom Architecture

The dichotomous aim of a custom CNN architecture was to both have a simple, intuitively understandable model, and a model easily trainable from scratch that nevertheless achieves a high accuracy. For specifying the custom network's architecture to tackle this problem, we took multiple aspects into consideration:

Firstly, the images show a high level of intrinsic conformity. One one hand, a scratch only needs a small amount of the entire image information to be classified as a scratch, as its information is very locally encoded. On the other hand, crazing is distributed over the entire image, showing self-similarities in different image parts. Thereby, we concluded that surface defect information do not depend on small-scale image information. To account for that, filter sizes of 15 and 11 px are used during convolution, much bigger as the commonly used $3-5$ px, for capturing image information on a larger scale. Additionally, we introduced max-pooling for increasing the dense neuron's receptive field and simultaneously reducing the model's parameters and thereby complexity. Max-pooling should not hinder the model's performance due to the observed self-similarity.

Secondly, the surface defects are not solvable with low-level features. This might not be true for scratches, which might be detectable using simple edge detection, but the other defect types show a inter-class similarity that is too complex for using only a single convolutional step. Therefore, we opted for the smallest architecture, that still allows us to obtain higher-level features from images: Two iterations of convolution and max-pooling, followed by a dense feed-forward network used for classification of the given six defect types.

The networks architecture is visualized in Fig. 4 and its detailed specifications are listed in Table I.

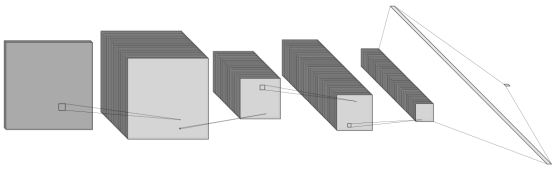

Figure 4: Visualization of the custom CNN's architecture. Two iterations of convolution and max-pooling are used, thereby, the channel information gets spread out from one to 64 channels in total. The convolution's output is used for classification using a dense feed-forward network. Detailed specifications are given in Table I. Created using [10].

For its advantages during training from scratch, BatchNorm, Dropout and Data Augmentation were used (see section III-A). The model's parameters - e.g. dropout rate, channel depth - were obtained by applying different educated guesses and then choosing the best-performing parameter set.

Operation	Depth	Size [px]	Filter size [px]
2D Conv	32	184	15
Max-Pool	32	92	2
ReLU + BN	32	92	-
2D Conv	64	82	11
Max-Pool	64	41	2
ReLU + BN	64	41	-
Flatten	$64 \cdot 41^2$	-	
Dense	512	-	-
Dropout $p = 0.3$	-	-	
Dense	6	-	-

Table I: Detailed architecture of the custom CNN. Channel depth and size information are specified after each operation.

C. VGGs

For using pre-trained models, we resorted to the group of CNNs developed by the Visual Geometry Group (VGG) in Oxford [11]. These group of networks were state-of-the-art image classification models on ImageNet in 2014. Their advantage over AlexNet was that they did not use large filters, but instead stuck to smaller filters of size 3 px by simultaneously making the network deeper. Thereby, the network was able to create higher-level abstractions during image processing, which improved their accuracy.

There are four common types of VGGs: VGG 11, 13, 16, and 19. They were pretrained on ImageNet iteratively by starting with the smallest net with 11 layers, then adding new layers in between, until the biggest net with 19 layers were reached. All VGG networks consist of multiple iterations of multiple convolutions followed by max-pooling. In this process, channel information get split up to 512 channels. There are also BatchNorm variants of the VGG nets, but as we're using pre-trained models and only fine-tune them, we chose to not use them, as their increased complexity does not justify the gain.

Today, VGGs are not state-of-the-art anymore [12], but due to their ready-to-use pre-trained implementation in PyTorch we chose them out of practical reasons.

1) VGG 16: As many suggested papers proposed (see [13] and section II-D), we started by implementing a pre-trained VGG 16, whose architecture is shown in Fig. 5. Even though it is one of the bigger VGGs, it is still small compared with current state-of-the-art networks.

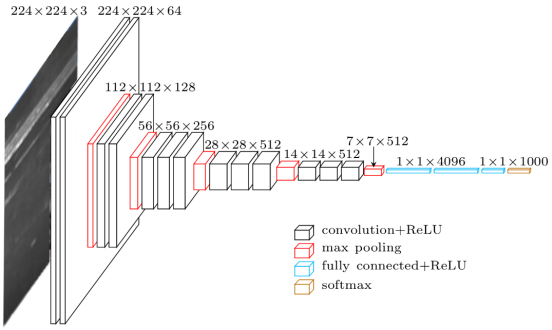

Figure 5: Visualization of the VGG 16 architecture. For specifications see [11]. For our purposes, we modified the last classification layer to only six instead of 1000 classes. Taken from [14].

Because this network was able to achieve high accuracy (see section IV), we were concerned with the amount of parameters - and thereby computational resources - the network uses. Therefore, we wanted to test whether a smaller VGG could reach a similar accuracy.

2) VGG 11: The only difference between the smaller VGG 11 compared to the VGG 16 are fewer convolutional layers. Its architecture is similar to Fig. 5, only missing one convolutional layer right before every max-pooling. Due to its smaller size, the VGG 11 needs fewer parameters and thereby reduces computational effort and time. It is the smallest pre-trained network available, therefore, the most efficient way of using transfer learning on the NEU surface defects.

For adapting it to the six defect classes, the last classification layer was replaced; then the entire network was fine-tuned.

D. Multi-Label Classification

The original NEU surface defect database for classification consists of images with only one defect type for each sample. However, in practice, an image may contain multiple defects of the same or different types. To take this into account we need to solve multi-label classification task which is a generalization of multiclass classification.

The additional *NEU_DET New* dataset with 1294 images was provided for evaluation of the results for multi-label classification. New dataset uses 4 samples from original NEU surface defect database to create a new image, an example is shown in Fig. 6. The label for resulting images can be represented as a vector. For example, vector [2, 1, 1, 0, 0, 0] denotes a sample where the first defect type appears twice, the second and the third defect types appear once, while the other defect types are absent.

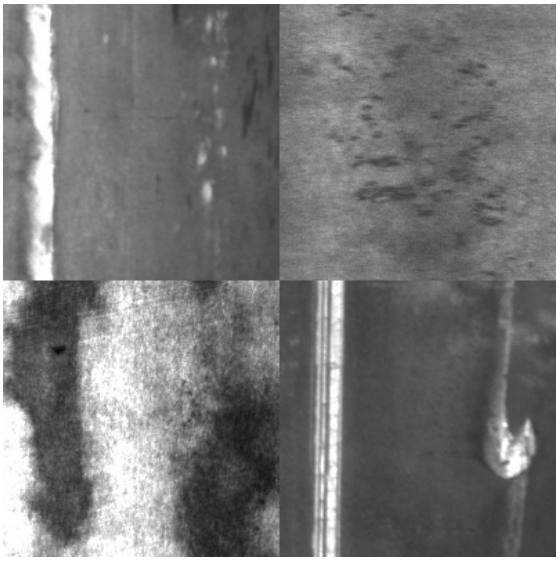

Figure 6: Sample from the dataset for multi-label classification.

In the same way as before, we used pre-trained models VGG 11 and VGG 16 for classification, changing the last layer of the network in this case. The weights of a backbone network can be updated during the process of training, we tried both options with and without "frozen" weights. The values of the resized output layer were rounded to fine-tune the model and get the final result. To train the model we used mean squared error loss function (MSE), which calculate the difference between the actual label and the output vector. It is worth mentioning that there are no examples with exactly the same positions of the defect types in the provided dataset. That is why data augmentation using rotated images significantly improved the performance of the model.

IV. Testing Results

For training and testing a stratified train-test-split of 70/30 with a batch size of 50 was used. Thereby, each training epoch used 26 batches, yielding an augmented dataset size of $26 \cdot 50 \cdot 4 = 5200$ images. Therefore, testing was performed on the remaining 2000 images. The cross entropy loss function then was optimized using Adam [15], which is using advanced methods to overcome shortcomings of regular stochastic gradient descent, and a learning rate of $\mathrm{lr} = 10^{-4}$. As activation functions ReLUs were chosen as they lead to a better convergence and not suffer from vanishing gradient. Each network's input are 200×200 px surface defect images. Their output varies for single-label and multi-label classification: Either the most likely label is chosen or the multi-class vector gets rounded to integer values. All training and testing was performed on

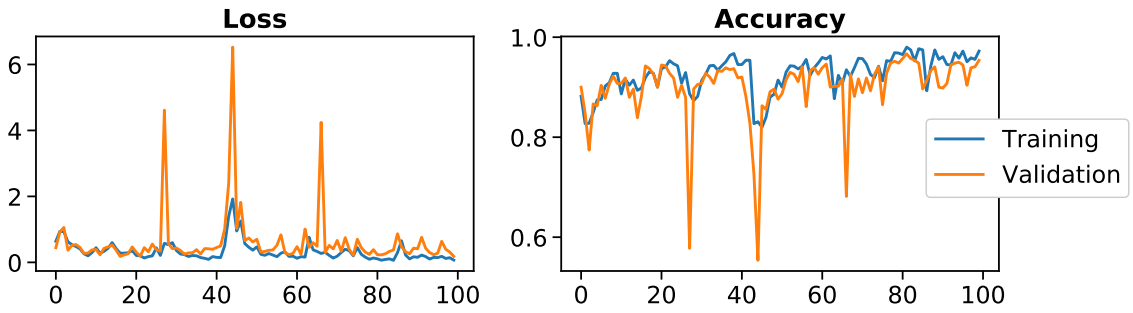

Figure 7: Single-label learning curve of the custom CNN. The network is converging after around 100 epochs and yields an accuracy of up to 97.7 %.

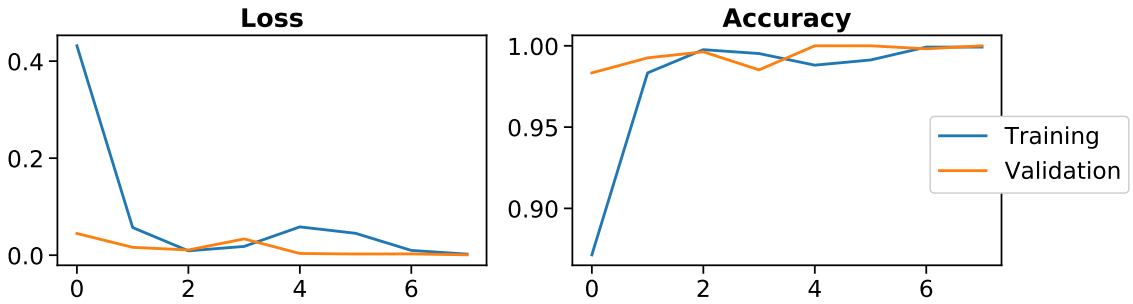

Figure 8: VGG16's single-label learning curve. After only 7 epochs no misclassifications occur. Each epoch needed 17 seconds during training time in our environment.

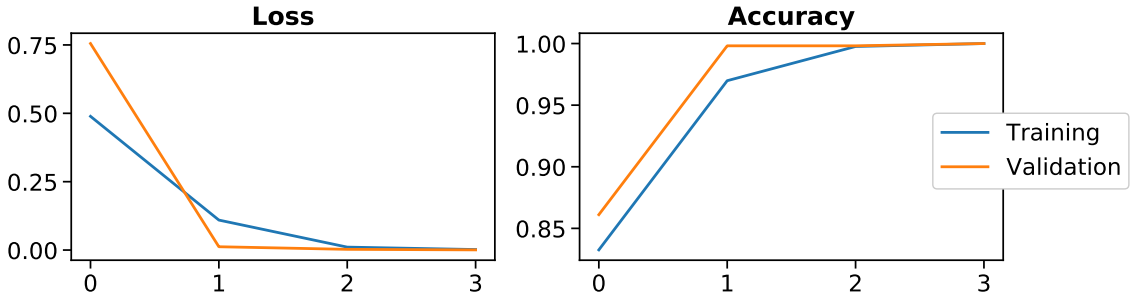

Figure 9: VGG11's single-label learning curve. The network converges in even less epochs than the VGG16 to an accuracy of 1.0. Additionally, during training each epoch needed 10 seconds, nearly cutting VGG16's training time per epoch into half.

a Tesla P4 GPU using Google Colab. To analyze learning rates and accuracy, one network per approach was trained. If it reached a training and test accuracy of 99.9 %, equivalent to no occurring misclassification, training was stopped. In the following, the different approaches will be analyzed and compared to each other.

A. *Single-Label Classification*

1) Custom CNN: As seen in the learning curves of the custom network (Fig. 7), the network was trained from scratch and required up to 100 epochs to converge. Even though each training epoch only needed 5 seconds on average, the custom network's training took the most time and computational effort due to the increased number of epochs.
Several spikes in loss and accuracy curves indicate that the network is not done learning and hint to local minimum outbreaks during optimization. After more than 30 epochs without such event, learning was stopped at 100 epochs as there the network showed signs of convergence. Additionally, during other training sessions, networks trained for more epochs showed no increased accuracy after exceeding 100 epochs.
It should be noted that the network is not overfitting, as training and validation curves do not diverge. In chapter V, further ideas on how to increase accuracy are discussed. As we focused on

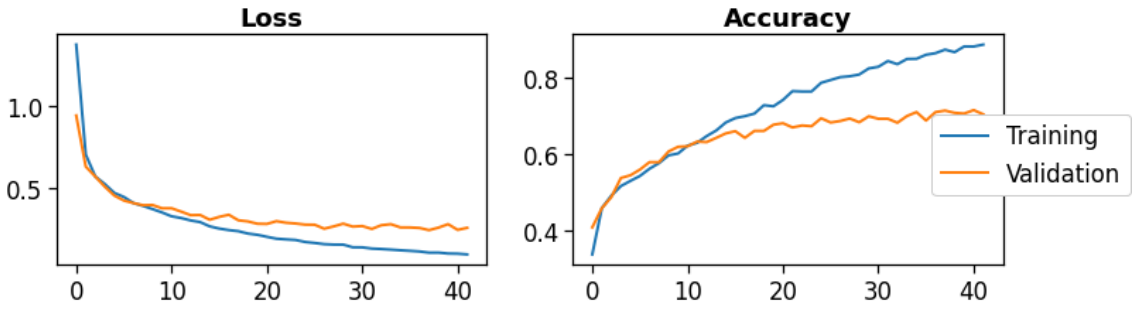

Figure 10: VGG16's learning curve for the multi-label classification task. The data augmentation was not performed and the model experienced overfitting.

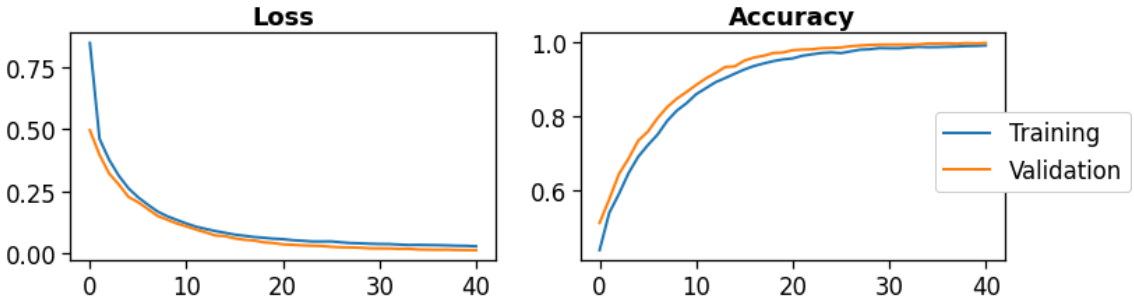

Figure 11: VGG16's learning curve for the multi-label classification task. This time data augmentation was performed and model achieved 100% accuracy after 39 epochs

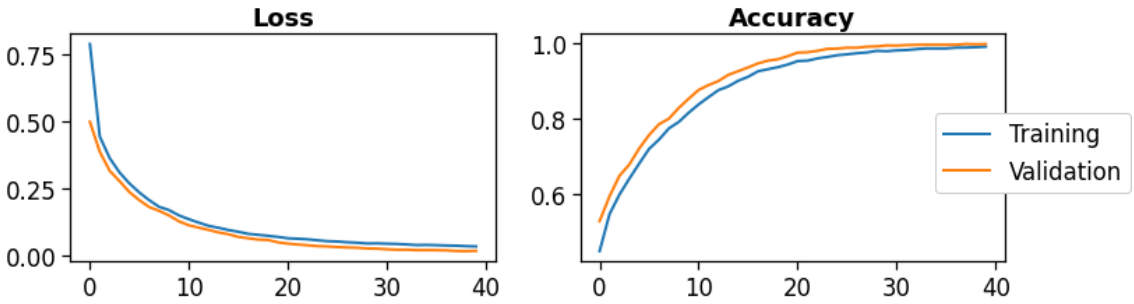

Figure 12: VGG11's learning curve for the multi-label classification task. This model converged faster than the model with VGG 16

comparing different approaches, we did not fine-tune our model. Nevertheless, advances methods are expected to lead to an even higher accuracy, edging closely to perfection. After 100 epochs, the custom CNN achieved an accuracy of 97.9 %. Types of misclassifications are specified in Appendix A.

2) VGG16: The VGG16's learning curve indicates a rapid convergence (Fig. 8), needing seven epochs until convergence. Due to the higher complexity and number of parameters compared to the VGG11, the 17 seconds training time per epoch is nearly twice as high as the simpler VGG's. After seven epochs, the VGG16 reached an accuracy of 1.0, classifying all images of the test and training data set correctly.

3) VGG11: The simplest version of pre-trained VGG networks converged even faster than the previous networks (Fig. 9), needing only three epochs until perfect convergence. Due to its lower complexity, each epoch during testing took 10 seconds in the specified environment.

B. Multi-Label classification

For multi-label classification task we also used a stratified train-test-split with a batch size of 50. However, this time 80 % of data was used for training and 20 % for validation.

During the evaluation the model suffered from overfitting after a few epochs. This problem led to a notable 15 % difference in accuracy between

training and validation stages. After 40 epochs and approximately 12 minutes of real-time our model achieved 91 % of accuracy which is significantly lower than the results for single-label classification. The results without data augmentation are shown in Fig. 10.

We assumed that the main reason for this is the way of constructing NEU_DET New dataset and splitting the data to training and validation sets. In this case datasets do not contain any matching labels. In addition, we do not simply check the presence or absence of some defect type but also calculating exact number of occurrences for each class.

In order to provide the data augmentation and more balanced training we used rotated images. Experiments showed that the performance of the model improved with the newly generated data and accuracy reached 100 %. Fig. 11 shows the learning curves for a model with VGG 16 as a backbone network.

We compared these results with the performance of VGG 11 model, which is shown in Fig. 12. As it turned out VGG 16 does not outperform VGG 11. Moreover, the model with VGG 11 is less time consuming. It took 811 seconds for VGG 16 to converge, while VGG 11 converged after 744 seconds.

V. Conclusion

Summarized, each of the three different approaches highlights interesting aspects of image classification on the NEU surface defect dataset. The different network's key performance parameters are listed in Table II.

	Custom	VGG16	VGG11
Accuracy	0.979	1.0	1.0
TTpE	5	17	10
EuC	100	7	3

Table II: Key performance indicators of the three compared CNNs. Both VGG's reached the highest possible accuracy, but differ in the training time per epoch (TTpE). These differences are explained in the different number of parameters in both networks. Even though the custom CNN required the smallest TTpE, the training process still took the longest time as the amount of epochs until convergence (EuC) is much higher.

The relatively high accuracy of the very simple custom CNN indicates that even simple CNN architectures are sufficient for classifying the given dataset. More complex architectures achieve higher accuracy, but therefore need more time and computational effort during training. The performance indicators suggest that transfer learning using a very simple CNN might be the optimal solution given the NEU surface defect dataset with regard to training time and accuracy.

Moreover, we did not exploit all possibilities of improving our custom CNN. To achieve an even higher accuracy, we therefore discuss in the following some advanced techniques.

During testing, we only considered one network per approach. One common technique to improve performance is to use ensembles of networks. Given our custom CNN, which took a long time for training, we therefore suggest to take snapshots of the model after a fixed number of epochs and average them at the end of training. The advantage of this approach is that it does not need additional computational resources to achieve higher performance, as the model still only gets trained once.

Additionally, we did not systematically explore the hyper parameter space of different learning rates, optimizer types, and train-test-split ratios. Even though such exploration can be fairly time-consuming, finding optimal hyper parameters always increases accuracy.

Taken together, the widely used VGG16 yields perfect accuracy, but given the other CNNs simplicity and unused potential, it might be considered unnecessarily complex. This might hint to the existence of small, yet highly performant CNNs. Such networks could be used as backbone networks for defect detection or region proposal networks. The consequential potential of such networks to reduce computation effort and therefore valuable time in real-time applications, make further exploration of such networks compelling.

Eventually, we would like to address the quality of scientific papers using this dataset. Up to very few exceptions, all papers lack of basic scientific requirements. Very similar vocabulary and formulations are used, research constrains on re-stating already known facts, the lack of English grammar is astonishing and on top of that, nearly all citations form a cluster of Chinese scientists, which re-cite each other and claim ownership of invention of CNNs and transfer learning. Based on these major shortcomings, we would strongly advice to not use this dataset as foundation for student projects. Other image classification datasets like Fashion-MNIST [16] or iNaturalist [17] exhibit both more challenging characteristics and higher-quality citations.

References

[1] K. Song and Y. Yan, "A noise robust method based on completed local binary patterns for hot-rolled steel strip surface defects," *Applied Surface Science*, vol. 285, pp. 858 – 864, 2013.

[2] H. Dong, K. Song, Y. He, J. Xu, Y. Yan, and Q. Meng, "Pga-net: Pyramid feature fusion and global context attention network for automated surface defect detection," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2019.
[3] C. Wang, Y.-t. Liu, Y.-n. Yang, X.-y. Xu, and T. Zhang, "Research on classification of surface defects of hot-rolled steel strip based on deep learning," *DEStech Transactions on Computer Science and Engineering*, 2019.
[4] Y. He, K. Song, Q. Meng, and Y. Yan, "An end-to-end steel surface defect detection approach via fusing multiple hierarchical features," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 4, pp. 1493–1504, 2020.
[5] "A comparative study of supervised and unsupervised learning algorithms for surface defect detection," https://git.io/JfGHD, 2020.
[6] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," ser. Proceedings of Machine Learning Research, vol. 37. PMLR, 07–09 Jul 2015, pp. 448–456.
[7] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, "How does batch normalization help optimization?" Curran Associates, Inc., 2018, pp. 2483–2493.
[8] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, 2014.
[9] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *CoRR*, vol. abs/1207.0580, 2012.
[10] A. LeNail, "Nn-svg: Publication-ready neural network architecture schematics," *Journal of Open Source Software*, vol. 4, no. 33, p. 747, 2019.
[11] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.
[12] "Comparison of image classification methods," https://paperswithcode.com/sota/image-classification-on-imagenet, accessed: 2020-03-16.
[13] Z. Lin, Z. Guo, and J. Yang, "Research on texture defect detection based on faster-rcnn and feature fusion," in *Proceedings of the 2019 11th International Conference on Machine Learning and Computing*, ser. ICMLC '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 429–433. [Online]. Available: https://doi.org/10.1145/3318299.3318341
[14] "Blog post vgg in tensorflow," https://www.cs.toronto.edu/~frossard/post/vgg16/, accessed: 2020-03-24.
[15] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.
[16] H. Xiao, K. Rasul, and R. Vollgraf. (2017) Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.
[17] G. V. Horn, O. Mac Aodha, Y. Song, A. Shepard, H. Adam, P. Perona, and S. J. Belongie, "The inaturalist challenge 2017 dataset," *CoRR*, vol. abs/1707.06642, 2017. [Online]. Available: http://arxiv.org/abs/1707.06642

APPENDIX A
CONFUSION MATRICES

The custom CNN misclassified after training until convergence $2.1\,\%$ of test images. The associated confusion matrix (see Fig. 13) helps understanding the characteristics of these misclassifications. Most mistakes occur for inclusion defects: $8\,\%$ of predicted inclusions were actually pitted surfaces, while $2\,\%$ of actual inclusions were predicted as scratches. These findings are explainable as inclusions are very similar to pitted surfaces and scratches on low representation level.

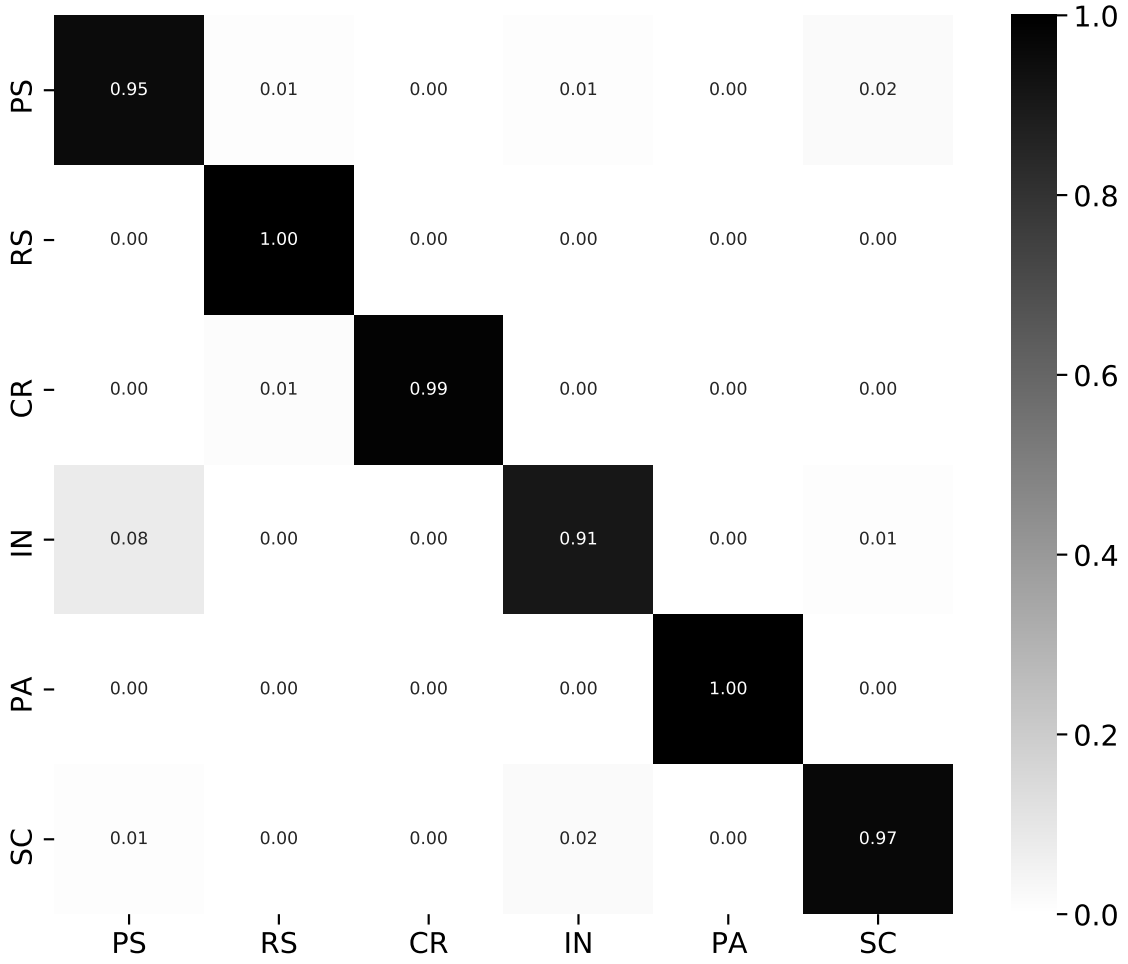

Figure 13: Confusion matrix of the custom CNN. Most misclassifications occur related to the inclusion defect type.