WELD CLASSIFICATION FOR DEFECT IDENTIFICATION

Abhijit Nayak
Data Science, Luddy IUB
Bloomington, IN
nayakab@iu.edu

Abhisek Panigrahi
Data Science, Luddy IUB
Bloomington, IN
abpani@iu.edu

Pratap Roy Choudhury
Data Science, Luddy IUB
Bloomington, IN
prroyc@iu.edu

Shubham Mohapatra
Data Science, Luddy IUB
Bloomington, IN
shmoha@iu.edu

***Abstract*—A lot of industries use automated welding machines to complete welding jobs. These automated processes can weld automatically for a lot of hours at a single stretch. This does not require any human intervention. But despite this exciting invention of autonomous welding, there are possibilities of the occurrence of defects in the welding done by these machines. Hence it becomes necessary for these defects to be identified immediately. Our work is to develop an image classification model which would efficiently identify whether the welding done by the automated machines is good or defective.**

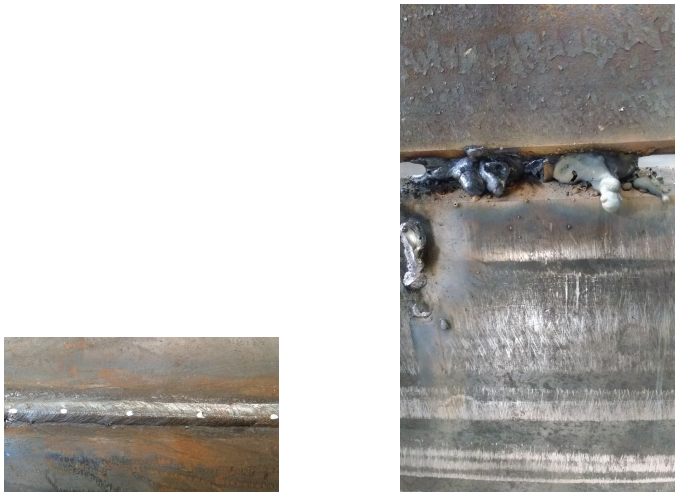
***Index Terms*—image classification, wedling classification, computer vision, neural networks, transfer learning**

I. Introduction

Image Classification refers to the task of identifying, what categories the images belong to. The categories/classes could be binary or multiple classes as well. The task of identifying the welding done by machines whether a certain welding is good or bad is called welding classification for defect identification. Good welding is easy to distinguish which will be straight and uniform with no slag, cracking, or holes. Whereas bad welding lacks uniformity, is too thin, and cracks down the middle of the bead. These are often caused by improper techniques or parameters. This project is interesting in terms of automating the identification of defects in welding materials as in any architectural industry. Due to the huge manual workload, it is often difficult to do a proper quality check on the metal welds. So images of any welding on the structure or its components can be easily classified as good or bad as quality measures with the help of modern image processing and deep learning techniques. This can save the manual check and time of any construction process and reduce the human error that can cause construction defects, loss in industrial work, monetary loss, etc.

We can see in the images below cases of suitable welding and bad welding, we can see an example of very bad welding which the human eye can easily detect.

However, there are more cases in which the differences are more subtle which we need to look at more clearly. Therefore our model needs to be trained in a manner that it can detect all these different types of nuances to detect bad welding types.



(a) Good Weld (b) Bad Weld

Fig. 1: Welding types

II. Related Work

In this paper [2], they have implemented deep learning techniques such as a customized DCNN which used some pre-trained layers of transfer learning models like GoogleNet, ResNet50, VGG-16, VGG-19, and ResNet101. They used X-ray images as their data set to train the models on and performed some pre-processing/augmentation techniques like cropping, and re-scaling on the original data. They have 1 input layer, 5 convolutional layers, 2 fully connected layers, and 1 output layer in their image classification model. Finally, they compared their model's result with that of the popular pre-trained networks.

III. Methodology

We implemented image classification models using large pre-trained convolutional neural networks like ResNet50, and ShuffleNet V2 along with different optimization techniques (Adam, SGD with momentum, RMSprop, AdaBound) to achieve the above-mentioned objective. Before modeling, we also performed some image augmentation techniques like

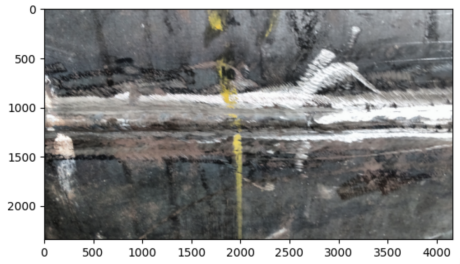


Fig. 2: Original Image

resizing, rotating, flipping, and introducing randomized brightness and contrast values using the Albumentations package. This step helped to balance the train data set and create more images (variations) which would help the models in generalization. Four different optimizers were used with both the pre-trained models in order to compare the convergence and generalization rates. Finally, we verified our model's performance with some evaluation metrics at places like accuracy and compared average training time per epoch, and training loss for measuring the performance in terms of better convergence and generalization.

A. Data Loader

ImageFolder class from torchvision was used to create train and test loaders for training and testing the models.

B. Image Augmentation

Image augmentation plays a vital role in image classification, segmentation, and detection; balancing the classes of the dataset and adding more random variations in the training dataset to help the models generalize. In this project, we used a popular Python library called Albumentations to perform resizing (128x128) from the initial data size (4160x2340), rotation, horizontal flip, vertical flip and randomized contrast, brightness using the ColorJitter method. We made sure that no repetitions were occurring in the augmentations of flipping, rotation, brightness, and contrast by introducing a probability of 0.2 and 0.3 (in horizontal flip) but distributed equally.

Pseudo Code:

transform :-

Resize (width = 128, height =128, probability = 1.0),

Rotate (-90, +90),

Horizontal Flip (probability = 0.3),

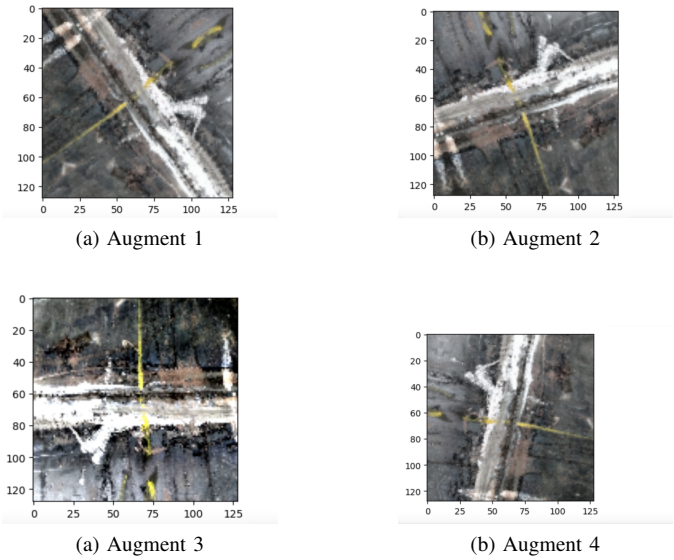
Vertical Flip (probability = 0.2),

ColorJitter (contrast=2, probability=0.2),

ColorJitter (brightness=2, probability=0.2),

Normalize (mean=0, standard deviation = 0, max-pixel-value=255)

We can see from the images below the augmentations of one single image in Fig2. We created good augmentations for 30 iterations and 125 iterations for bad augmentations.



(a) Augment 1 (b) Augment 2

(a) Augment 3 (b) Augment 4

Fig. 3: Augmentations

C. Modelling

Here, we chose to implement deep learning especially large sized convolutional neural networks to carry out our task because it helps in automatically extract features from images in a better way compared to the traditional machine learning techniques. CNNs are a replica of a human visual cortex system, so these networks work proficiently in image related tasks. In this project, we trained such large convolutional neural nets to take out important areas or region of interest from our welding images. These region of interests could be those based on the good and bad welding aspects like uniformity, no cracks and non-uniformity, cracks, too thin respectively. After training those models, they were able to automatically classify on unseen images whether they were good or bad welding.

1) Transfer learning: Transfer learning is an advanced method where a pre-trained model gets reused to carry out another task. These pre-trained models would have been trained on some other large amount of data and these optimized weights are then used to perform various tasks like classification in this case. Because carrying out the other task might not have enough data to start with, then in that case these pre-trained optimized weights would play a vital role in performing that other task with greater efficacy. Our choice of models were ResNet50 and ShuffleNet V2. We customized the final layers of these 2 pre-trained models in order to address the problem statement at hand. We changed the number of output features in the final layer to 2 as we were dealing with a binary classification problem and it would give us the probabilistic scores in the output.

2) Optimizers:

a) Adam: It inherits the properties of Adagrad, RMSProp and SGD with momentum. It uses RMSProp to use the squared gradients to scale the learning rate and also SGD with

momentum to accelerate convergence in the right direction and facilitate good generalization.

b) SGD with momentum: Momentum is faster than SGD so SGD with momentum will converge faster as it helps in accelerating the gradient vectors in the right direction and also increase training accuracy.

c) RMSprop: RMSPRop is an upgrade from Adagrad which uses the second moment to speed up the training process. It changes the scale of our learning rate so that the algorithm can take larger steps converging faster.

d) AdaBound: It was introduced in 2017. It introduces the concept of clipping in learning rates to apply a dynamic bound on them. So that at start the optimizer behaves with adaptive learning rate properties just like Adam, Adadelta etc. but gradually moves towards acting like SGD to facilitate good generalization of the data.

3) Models:

a) ResNet50: ResNet50 is one of the very popular deep learning pre-trained frameworks used in various computer vision related tasks. It is also known as Residual Network with 50 deep neural layers. Bottleneck building block design is used in this network which helps reduce number of matrix multiplications and parameters. It was introduced in 2015 and was trained on ImageNet database. The architecture contains of a 7x7 kernel convoluition, max pooling layer and 48 convolutional layers summing up to 50 layers. The final layer has an average pooling layer and a fully connected layer with 1000 nodes with a softmax activation function [4],[5].

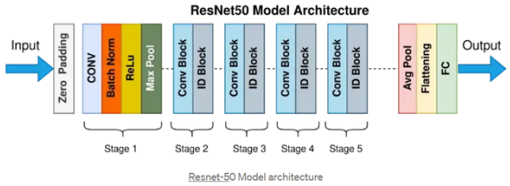


Fig. 4: ResNet50

Another important aspect of ResNet50 is that it tackles 2 important problems in deep learning: (a) exploding/vanishing gradients and (b) degradation by the introduction of skip connections. It basically skips a bad performing layer and jumps to the next layer and hence prevents the deterioration of performance of the network. ResNet50 was trained with Adam, RMSprop and SGD with momentum optimizers.

b) ShuffleNet V2: The ShuffleNet V2 was introduced in 2018. This network is built upon ShuffleNet v1 which was introduced in 2017. The main building blocks of ShuffleNet v1 are group convolution, depth wise convolution and channel shuffle. In group convolution there are group of convolutions which reduces the computation time and also requires less computing power. Depth wise convolution used for decreasing the computation time for convolution. We can take an example

Fig. 5: ShuffleNetV2

of convolution operation involving 3 color channels. During convolution we can split the color channels to 3 separate and then convolution on them separately. It decreases the matrix multiplication by thousands of operations. Also, channel shuffle helps in relating to the past features. In ShuffleNet V2 a new concept was introduced called channel split where an input channel is split into 2 halves and the 2nd half is fed to the next block to facilitate feature reuse [6]. ShuffleNet V2 was trained with AdaBound optimizer.

IV. Experimental Setup

Here, we will be talking briefly about the software versions, dataset, hyperparameters used in this project and the hardware platforms.

TABLE I: Table of software version details

Softwares	***Versions***
matplotlib	3.5.2
numpy	1.23.1
torch	1.12.0+cu113
torchvision	0.13.0+cu113
opencv_python	4.5.5.64
tqdm	4.64.0

Hardware used for this project was a Windows 10 laptop 64 bit, with 16GB RAM, 6GB Nvidia RTX 3060 GPU and an i7 11th Gen processor.

TABLE II: Table of hyperparameter details

Hyperparameter	***Value***
num_classes	2
learning_rate	0.01
batch_size	32
num_epochs	20
weight_decay	0.0001
momentum	0.5
betas	(0.9, 0.999)
gamma	1e-3
eps	1e-8

The dataset is collected from Kaggle. It consists of 2 folders: train and test, each containing images of good and bad welding. The train set has 181 good images and 28 bad images. Whereas the test set has 15 good and bad images

respectively. The dataset is quite unbalanced initially. Source: https://www.kaggle.com/datasets/prataproychoudhury/weld-classification.

V. Results

A. Loss Function

We chose Cross Entropy as the loss function for this problem statement. As it has the ability to evaluate how well our classification models perform in categorizing the images into good welding or bad welding (in this project). Cross Entropy loss is also able to evaluate categorization of data points into multiple classes as well.

B. Why batch size 32?

We had a lot of training examples after the augmentation process. There were 8930 augmented training images created from the 209 training set. For the ease of model training and optimization purposes, we took 280 minibatches for the entire augmented image set which creates each batch of size 32. In order to make the training process faster and to make sure that the cuda does not run out of memory we decided to choose such a smaller batch size.

C. Why learning rate 0.01?

We chose learning rate 0.01 for all the optimizers as the amount of training images were very high and very low learning rates were not recommended. Also, it helped us compare the performance of the models with different optimizers.

D. Plot of Training Losses for all models and optimizers

This plot clearly describes the performance of the model with optimizers combination. As we have 2 models ResNet50 and ShuffleNet V2 with four optimizers, it would required 8 different combinations. But for execution complexity and memory restrictions, we tried Resnet50 with Adam, SGD with Momentum and RMSProp, and ShuffleNet V2 with AdaBound.

To plot the training loss for all the images, we stored the average running loss for every 70 minibatches, that makes 4 training losses per epoch. As we have 20 epochs to run, therefore the plot shows the training loss for 80 epoch points in total.

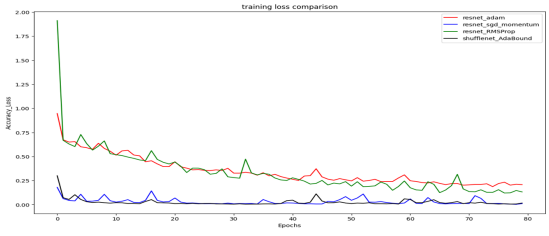


Fig. 6: Training Losses Plot

ResNet50 with RMSProp has the highest training loss (1.9) in the beginning but drops after 5 epochs, and eventually comes down to the nearly 0.132 after 80 epochs. ResNet50 with Adam optimizer performs similar to the Resnet50 with RMSProp but endd up with higher training loss of 0.208 after 80 epochs.

ResNet50 with SGD momentum optimizer shows better performance as starts from 0.179 and ends with 0.01 training loss. Whereas ShuffleNet V2 with AdaBound performs similarly but has a little more training loss. It starts at 0.3 but ends with 0.016 training loss after 800 epochs. Moreover, as AdaBound is combined form of SGD and Adam, it executes faster than all other optimizers and gives better test accuracy on image classification. It convergence rate is faster than experimented optimizers for Resnet50 models and gives best result in ShuffleNet V2 with AdaBound optimizer.

E. Inferences with Images

Once the model is trained and the best optimizer is chosen, it is then used to predict the class labels of the test images of welding materials. The below images depict the predictions from our model whether the images are good welding or bad welding on test images.

(a) Good

(b) Bad

Fig. 7: Model Predictions - Good and Bad Welding

F. Results Discussion with Table

AdaBound on ShuffleNet V2 and SGD (momentum) on ResNet50 have the best convergence rate. For the same set of augmented image data with same hyperparameter combinations, we have tabualted the results from all the optimizers execution and their training loss, test accuracy, and average training time per epoch. ShuffleNet V2 with AdaBound has the fastest average training time per epoch and also it has the best generalization with a 100% test accuracy. Adam and RMSprop have the worst convergence and generalization rate.

TABLE III: Result comparison for each model and optimizer

Model	***Optimizer***	***Training loss***	***Test Accuracy***	***Avg. Training Time per Epoch***
ResNet50	Adam	0.2	83	50 sec
ResNet50	SGD (momentum = 0.5)	0.007	96	55 sec
ResNet50	RMSprop	0.14	73	1 minute
ShuffleNetV2	AdaBound	0.008	100	45 sec

Though ResNet50 with SGD Momentum and ShuffleNet V2 with AdaBound performs nearly similar, but for our classification problem with given image set and all other experimental setup, ShuffleNet V2 with AdaBound is the best model and optimizer as it takes the lowest average training time of 45

seconds and gives maximum test accuracy. ResNet50 with SGD Momentum performs just better in terms of training loss of 0.007 which is lesser by only 0.001 than the other and it has test accuracy of 96% with 55 seconds of average training time per epoch.

VI. Conclusion

Welding inspection and quality control is very important in construction works as defects, visual or otherwise can cause breakage and compromise the efforts of final job. Sometimes manual inspection due to human eye error can mislead the defect identification, but the automated process of defect identification is the modern technical way.

As we have used the above mentioned methods, We could achieve a very good accuracy with SGD with momentum and ShufflenetV2. We also saw that ShuffleNetV2 model with AdaBound optimizer had fastest training time, best convergence and generalization rate followed by ResNet50 with SGD (momentum) optimizer. Thus the best model selection leads us to have ShuffleNet V2 with AdaBound optimizer that gives almost 100% accuracy.

VII. Future Scope

We have tried out only four combination of the model and optimizers. But as we have seen that SGD with momentum is also a potentially good optimizer choice, ShuffleNet V2 with SGD Momentum and other optimizers can be tried out to have a complete performance measure of all these models and optimizers.

Also, here we are just classifying the weld image as good or bad, but what kind of defects are there! That remains a mystery. The future work can include training more defective weld images with type of defect labels such as excess of overmetal, incomplete filling in shape and size defects, identifying cavities, finding hot and cold cracks. That can give the inspection work more precise automated results of type of defects with highlighted detected portion in the image.

VIII. Team Member Contribution

Our team collectively worked really hard and successfully delivered the proposed task.

Pratap has worked on selecting and ideation of the project work. He and Shubham worked initially on on the project proposal, data collection, setting up the image folder and directory. They developed the augmentation code snippet and created the augmented images, label encoded images for intermediate experiment purposes. They also prepared the final project presentation and arranged the results from the models.

Abhisek and Abhijit worked on creating data loader sets, training the neural network models, selecting the optimizers, plotting the training losses and and selecting the best model with optimizer.

All the members worked equally to create the report content based on their respective parts. Pratap and Abhijit prepared the final report, did all the validation check and required formatting.

References

[1] Adrian Rosebrock, “PyTorch image classification with pre-trained networks.” July 2021

[2] Chiraz Ajmi, Juan Zapata, Sabra Elferchichi, Abderrahmen Zaafouri, Kaouther Laabidi, “Deep Learning Technology for Weld Defects Classification Based on Transfer Learning and Activation Features.”, March 2021

[3] Emily Potyraj (Watkins), “4 Ways to Improve Class Imbalance for Image Data.”, March 2021

[4] https://datagen.tech/guides/computer-vision/resnet-50/

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun,” Deep Residual Learning for Image Recognition.”, December 2015

[6] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, Jian Sun,” ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design.”, July 2018