A Convolutional Approach to Quality Monitoring for Laser Welding

ECE 697 Summer2021 Final Project Report

Yeongeun Kim

Abstract

Laser Welding is a technique used to achieve successful hairpin processing in electric motor manufacturing. Alignment and tooling are the two primary difficulties encountered when laser welding hairpins. This project proposes a stable quality monitoring system for the laser welding process using an appropriate Convolutional Neural Network architecture. It uses the concept of VGG blocks in VGG models to extract features from the result images. It also explores overfitting prevention methods for better image classification.

Background History

In recent years, the market for electric vehicles has grown up at impressive rates. According to market research firm McKinsey & Co., the market – led by China – continues to grow at impressive rates. In this world of growing electrification, the demand for high-quality electric motors also continues to rise. To increase the energy efficiency of the electric motor, the electric vehicle manufacturers had to reduce the copper loss, which accounts for the biggest loss of the motor. Therefore, the hairpin winding techniques replaced the existing winding techniques, which were round wire winding. However, the hairpin winding leads to a high amount of contact points, and more careful concentration was required. Therefore, laser welding techniques were introduced, and auto inspection for the quality of laser welding has been required.

The present laser welding process consists of two steps; pre-welding and post-welding. Before welding, the detector should find each surface of hairpins apart from the background. Also, it should find the center of the surface, which would be the target point for the laser beam. After welding, the inspector classifies all the classes of welding. The general features of incorrectly welding are lack of fusion, adhered spatter, unwelded, blow out, over-welded, and misaligned. Based on the features from quality deviations, it should filter the incorrectly welded hairpins from the correctly welded ones. This helps to detect weld failures before they get further processed inline.

Problem description

There has been a rule-based algorithm and a machine learning algorithm in history. For the rule-based algorithm, they do not require a massive training corpus but cannot learn itself. In a while, the machine learning algorithm can learn itself, and with enough dataset, fast development is possible. In the case of laser welding, a rule-based programmed inspection for laser-welding might be able to recognize each state; correctly welded and incorrectly welded. However, sometimes, the same image can very well be considered as wrong. Therefore, we should consider a stable quality monitoring system for the laser welding process using the appropriate Machine Learning technique.

The use of Machine Learning algorithms to detect quality deviations in the welding of hairpins is still very retrograde. Furthermore, the accuracy for detecting different error classes based on simple 2D images is in a range from 61 to 91 % depending on each class. For industrial applications, this number is too small. Therefore, we should explore methods for increasing the accuracy.

Prior works

As Machine Learning has shown a lot of potential in recent years, there is already a decent amount of applications of Machine Learning techniques in laser welding. In [3] a system is proposed that uses a k-Nearest-Neighbor (kNN) method for weld quality classification based on spatter and plume images. In [4] artificial neural networks (ANN) are used for detecting cracks based on acoustic signals. A support vector machine (SVM) is used for defect detection of x-ray images in [5]. For the multiclass defect detection of radiographic images, both SVMs and ANNs showed good results in [6]. Furthermore, the suitability of convolutional neural nets (CNN) for defect detection was proven in [7].

Related works_[1]

The prior studies show that there are many different technologies for the quality monitoring of laser welding with image processing being an accurate but cost-efficient solution. So far, the Machine Learning techniques have been partly applied to laser welding problems such as defect detection. In the production of electric motors, some AI-based methods were evaluated, but not yet for the laser welding of hairpin windings. Therefore, [1] evaluates different scenarios for using Machine Learning in the quality monitoring of laser-welded hairpins.

In [1], the author investigated two data sources; machine parameters and visual information acquired by a CCD camera. Firstly, the machine parameters were used to predict weld defects and the overall quality of a weld seam before contacting assessment. This is because the hairpin winding is influenced by not only the mechanical but also the electrical properties of each contact point. Secondly, using convolutional neural networks, the image data was analyzed. By the severity, different network architectures for assessing the weld quality as well as for classifying visible weld defects could be made. Finally, the paper showed a combined quality monitoring system consisting of a pre-process plausibility test as well as a post-process quality assessment and defect classification.

To evaluate the performance of Machine Learning models, the author used the most common metrics being (1) accuracy, (2) precision, (3) recall, and (4) F1 score.

$$Accuracy = \frac{TruePositives + TrueNegatives}{TotalNumberOfPredictions} \quad (1)$$

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \quad (2)$$

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives} \quad (3)$$

$$F1\ score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \quad (4)$$

Sometimes, a high accuracy which means high true positives and true negatives rates is not the primary target. To minimize one of the two errors, namely false-negative and false-positive, Recall and Precision metrics should be considered. For a low false-negative rate, the recall metric is used. In a while, for a low false-positive rate, the precision metric is used. When both errors matter, the F1 score is used as it combines both precision and recall.

The author concludes that using reinforcement learning techniques, the self-optimizing quality control system could be made. The system not only independently finds the optimal machine parameters and but also adapts them dynamically to external influences. Furthermore, depending on the severity of the burr and the insulation residues, the laser welding process can be run with adapted welding parameters and the defective part can be replaced before.

Related works_[2]

As already said in the problem description, the use of Machine Learning algorithms in laser welding of hairpin winding is still a little retrograde. Furthermore, the prior studies have analyzed the potential of Machine Learning for quality monitoring in the process by using simple 2D images with a low-cost camera. As a result, the accuracy for detecting different error classes based on the images was very low level. Therefore, [2] deals with the development of a new experimental setup with a 3D scanner, preprocessing of data, and building an appropriate Convolutional Neural Network architecture to increase accuracy.

Using Keyence's XR-HT40M 3D camera, it was able to digitize the welds in the form of a 3D camera. Thereby, the height information was used for the inspection process, which enhances the stability of the inspection. The calculation unit of the camera calculated an RGB typed image with a special color-coding from the 3D height information, which became an output.

The author clearly describes the goal of the procedure of data generation as generating the largest possible amount of data with good quality and under realistic conditions. After the data generation procedure, about 600 labeled images of hairpin welds for each class with different severities were produced. By combining rotation, shifts, and mirroring of the images, the data augmentation technique could be implemented.

Because the preprocessing step has a huge impact on the resulting performance of a classifier, the author developed a specific preprocessing pipeline. After calculating the height information of the RGB image produced by the 3D camera, the median height of the welded pin and the range above and below the median value could be calculated. By applying average pooling with a kernel size of 15x15 pixels and scaling the height into a range of 0 and 2555, the dimension of the height data could be reduced. After these steps, further normalization was implemented to achieve better results.

Finally, the author shows a Convolutional Neural Network model architecture for the quality monitoring system. The model is made up of four convolutional blocks. Each convolutional block consists of two convolutional layers, which have 3x3 pixels kernel size, and a Max-Pooling layer, which has 2x2 pixels size. Through the following convolutional blocks, the number of kernels was doubled. Also, before the ReLU-Activation at each layer, Batch Normalization was used. After the fourth and last convolutional block, a Global-Average-Pooling was added. This flattens the output of the last convolutional block. At last, a Drop-Out layer was used after the fully connected layer before the final dense layer. The author says these methods were used to get good generalizations and to prevent overfitting problems.

For the parameter variation of the model, the research tried to vary the normalization technique, number of Convolutional Blocks, and number of filters per Convolutional Blocks. And then, as a classical validation method, 5-fold Cross-Validation was implemented. This proved the CNN architecture was capable of reliably deciding whether a welding process can fail. Also, as a visual validation method, a saliency map visualized the importance of pixels for decisions. This showed the CNN architecture was also capable of detecting relevant features for all severities and classes.

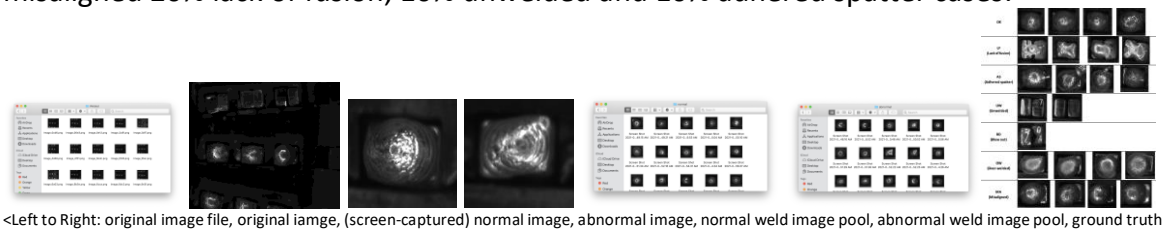
Topic Importance

Based on the above two reference papers, this project aims to make a stable quality monitoring system for the laser welding process using the appropriate Machine Learning technique. For the laser-welded hairpins, it also evaluates different scenarios in quality monitoring. Using convolutional neural networks, the image dataset would be analyzed. Different network architectures assess the weld qualities and classify the results by case. Finally, the common metrics to evaluate the performance of Machine Learning would show whether the model can be a stable system. To achieve high accuracy for detecting different classes based on the existing dataset, it also uses sophisticated technologies such as preprocessing the data and building an appropriate model architecture. This auto inspection for the quality of laser welding would help to detect weld failures before they get further processed inline.

Project Approach

Dataset

The original image file has 143 images and each image has 3 welding results. Therefore, I had to cut out each welding result from the original image to let one image show only one welding result. By matching the center of the capture screen with the center of the welding result, enough pixel information could be attained from the images. Also, by adjusting the capture screen size, all the images have the same size, 230 by 230. Based on a ground truth table chart, I classified the images into normal and abnormal cases. There are six cases of abnormally welding cases in the chart; 'lack of fusion', 'adhered spatter', 'unwelded', 'blow out', 'over welded', and 'misaligned'. When the final abnormal welding image pool was completed, it contains about 30% over-welded 30% misaligned 20% lack of fusion, 10% unwelded and 10% adhered spatter cases.



<Left to Right: original image file, original iamge, (screen-captured) normal image, abnormal image, normal weld image pool, abnormal weld image pool, ground truth table chart>

After making normally and abnormally welding image pools, a whole dataset had to be made. To convert the image pools into the dataset, a loading process was essential. Before start, I assigned the labels "normal" and "abnormal", which were the actual name of the original image pools. Plus, I set the initial image size like 224 to make a coherent image size. Using the OS library, each path with an assigned data directory location was accurately collected. Two classes' labels were indexed into numbers 0 and 1. Before preprocessing the data, using the OpenCV library, the BGR typed images were converted into the RGB typed ones. Also, using the same library, all the images were resized into the same size 224. Finally, all the identically sized RGB typed images were collected into a whole dataset which has two cases; normal and abnormal.

Preprocessing

With the dataset, I followed a basic data preprocessing flow. Using Keras train_test_split tool, the whole dataset was randomly split into 1:3 train and validation datasets. All the features and labels in the training dataset were assigned to X_train and y_train. In the same way, all the features and labels in the validation dataset filled the empty X_val and y_val arrays.

After the data generation, the data in each feature and label had to be normalized. In each train and validation feature, they were divided by a number 255. Also, they were reshaped into the same size, which was 224. This normalization step makes the image data have 0 to 1 pixel range and lets the further steps be more stable thanks to the consistent amount of data.

For data augmentation, I chose a simple tool from the Keras library. The Keras preprocessing module, named Image Data Generator, was used. By rotating, zooming, shifting, and flipping, the small amount of the data could be augmented. In the Image Data Generator, not only with the augmentation skills, users can set the mean of input and each sample to 0, divide them by each std, and do some whitening skills to prevent noise problems.

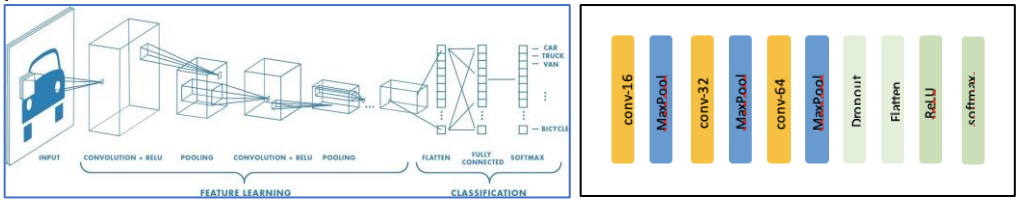
Model Architecture

For the Base-Line model architecture, I chose a Convolutional Neural Network (CNN) model. As a kind of deep learning algorithm, the network can successfully capture the spatial and temporal dependencies in the input image through some applications of relevant filters. The most advantageous role of the network is that it can reduce the images into a form that is easier to process, without losing critical features for getting a good prediction. To extract the features, the CNN consists of the convolutional layer and the pooling layer. Using the convolution operation, high-level features such as edges can be extracted. Using the pooling operation, the computational power required to process the data can be decreased through dimensionality reduction. Furthermore, it is useful for extracting the dominant features.

Using the Keras models and layers, a Convolutional Neural Network model architecture could be made. 4 convolutional blocks contain a convolutional layer followed by a max-pooling layer. All the convolutional blocks have the same structure. But the number of kernels were doubled, starting from 16.

After the last feature extraction operation, a drop-out layer was added. The drop-out is a kind of regularization method. It makes the training process noisy, forcing nodes within a layer to take on more or less responsible for the inputs. This means the dropout breaks up situations where network layers co-adapt to correct mistakes from prior layers, in turn making the model more robust. Therefore, we could call the dropout the simplest way to prevent neural networks from overfitting problems.

For a non-nonlinear activation function, I used the ReLU function. The ReLU activation function is famous for overcoming the vanishing gradient problem and allowing models to learn faster and perform better.



<Left to Right: (cited) Convolutional Neural Network basic architecture, my Base-Line model architecture>

Evaluation

To evaluate this model, I chose the Adam optimizer as an optimization method. Comparing to the classical stochastic gradient descent procedure, the Adam optimizer has attractive benefits on non-convex optimization problems. Not only with the gradient benefits, but the hyper-parameters can also be interpreted very intuitively with a little tuning. While the stochastic gradient descent maintains a single learning rate for all weight updates and during the whole training, the Adam adapts the parameter based on the average of both the first moment and second moment of the gradients. Specifically, the algorithm even calculates an exponential moving average of the gradient and the squared gradient. Finally, the Adam optimizer could be used in this quality monitoring system model thanks to its good results and fast speed.

Also, as a loss function, the Sparse Categorical Cross entropy was used. The loss function Sparse Categorical Cross entropy is different from the Categorical Cross entropy. Comparing to the Categorical Cross entropy, which requires data is one-hot-encoded, the Sparse Categorical Cross entropy can leave the numbers as they are. Therefore, for the pixel-ranged image dataset, I chose the Sparse Categorical Cross entropy as a loss function.

With a 1e-5 learning rate, I trained the model for 500 epochs since the learning rate was very small. Through the epochs times, the accuracy and loss per the iterating times' graphs could be printed. To see the final performance of the model, a classification report with some common metrics (accuracy, recall, precision, and F1 score) was achieved. Using the sci-kit learn library when all the prediction values of the model were accumulated, the classification report chart could be visualized.

Project Developments

VGG Net & VGG blocks

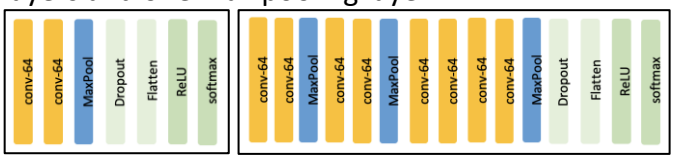
For better Image Classification, the Base-Line model architecture has been transformed. Many models have achieved great results for image classification. Each model has used its discrete architecture elements many times. The ResNet has the residual module, the GoogLeNet has the inception module, and the VGG blocks are in the VGG models. For one of them, I picked the VGG Net's VGG block as my model architecture's alternation.

The VGG Net, especially VGG16 was proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition". They achieved 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes.

The basic architecture of VGGNet has groups of convolutional layers that use small filters followed by the max-pooling layer. This means, thanks to the repeated 3by3 filters, it does not need to learn not so many numbers of parameters. Also, the 3by3 filter is the smallest size to capture every direction of a pixel image. In addition, as the network grows deeper, thanks to the non-linear function, the ReLU, the decision function becomes more discriminative.

Refer to the structure of the VGGNet, the specification of the VGG block can be generalized. There are one or more convolutional layers with the same number of filters and a filter size of 3by3, stride of 1by1, same padding. Therefore, the output size is the same as the input size for each filter, and it uses the rectified linear activation function. The following max-pooling layer has a size of 2by2 and stride of the same dimensions.

Finally, applying the VGG blocks, two types of models could be made. Changing the number of the VGG blocks, 1 VGG block model and 3 VGG blocks model have been evolved based on the Base-Line model. The 1 VGG block model has 1 VGG block which contains two convolutional layers (each has 64 # kernels, (3,3) pixel size, same padding) and one max pooling layer ((2,2) pixel size, strides=(2,2)). The 3 VGG blocks model has 3 VGG blocks and each block has a different number of kernels in the convolutional layers. They are doubled through the VGG blocks. The inner setting is the same as the 1 VGG block model. Uniquely, the last block has four convolutional layers and one max pooling layer.



<Left to Right: 1 VGG block model, 3 VGG blocks model>

To prevent the overfitting problem, two common methods were introduced. Batch Normalization and L1/L2 Regularization were also applied to the Base-Line model, 1VGG block model, and 3 VGG blocks model.

Overfitting Prevention _ Batch Normalization

The batch normalization is a technique designed to automatically standardize inputs to a layer in deep learning neural network. This not only improves the model but also allows it to converge faster. The batch normalization process consists of two steps; normalizing the input and rescaling of offsetting. Getting a mean and a standard deviation of the inputs, it subtracts the mean from each input while dividing the whole value with the sum of standard deviation and smoothing term. The smoothing term assures numerical stability within the operation by stopping division by zero value. At the rescaling offsetting step, the gamma and beta parameters are used for rescaling and shifting a vector that contains values from the previous operations. Through the training, these parameters would be learned enough. This Batch Normalization technique was applied to my CNN model architectures. After the convolutional layer and before the max-pooling layer, the Batch Normalization layer exists.

$$\mu = \frac{1}{m}\sum h_i \qquad \sigma = \left[\frac{1}{m}\sum (h_i - \mu)^2\right]^{1/2} \qquad h_{i(norm)} = \frac{(h_i - \mu)}{\sigma + \varepsilon} \qquad h_i = \gamma h_{i(norm)} + \beta$$

<Left to Right: getting mean of input, getting std of input, normalized input, rescaling and shifting parameters for previous values>

Overfitting Prevention _ L1/L2 Regularization

Another option to prevent the overfitting problem is Regularization. The regularization means adding a penalty when the model complexity increases. Specifically, the regularization parameter (lambda) penalizes all the parameters except intercept so that the model generalizes the data and would not be overfitted. Performing L2 regularization encourages the weight values toward zero but not exactly zero. In a while, performing L1 regularization encourages the weight values to be zero. Among these, I chose the L2 regularization method for my algorithm. Despite the high level of computational cost, I expected more accurate performance.

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right]$$

$$\min_\theta J(\theta)$$

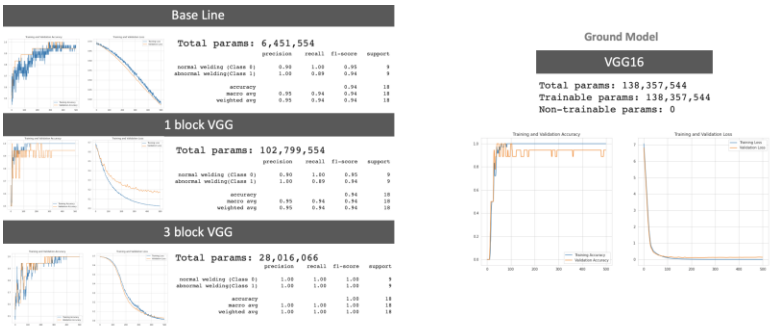
< Regularization in cost function, lambda penalizes all the parameters except intercept >

Project Results
Classification Report & Accuracy and Loss graphs

Finally, I analyzed the results. Comparing to the Base-Line model's performance, the 1 VGG block model and 3 VGG blocks model showed a little bit higher performance. Based on the classification report chart, the accuracy, recall, and precision metrics could be achieved.

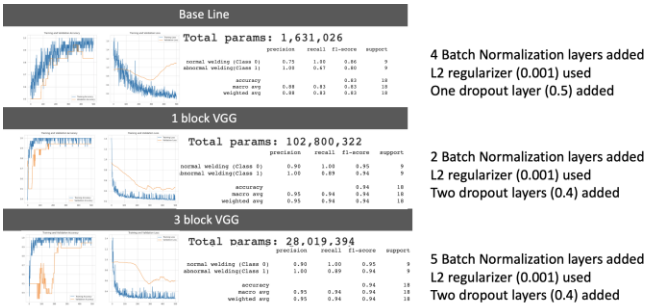
The total number of the parameters used in the network was the biggest in the 1 VGG block model, but still smaller than the ground truth model (VGG16)'s. Also, the accuracy and loss graph of the 1 VGG block model seemed to be very similar to the ground truth model. Generally, the number of parameters in a given layer has been regarded as the count of learnable elements for the filters in the convolution neural net. Therefore, it could be concluded that many parameters helped the 1 VGG block model to learn the image classification much more correctly.

On the other hand, the 3 VGG blocks model has a much deep layered VGG Net, so the number of parameters used for the performance was much smaller than the 1 VGG block model. However, through the repeated convolutional layers and max-pooling layers, it collected the better characteristics in the feature extraction. Therefore, the precision and recall metrics of the classification report showed all the numbers almost 1.



<Left to Right: loss and accuracy graphs and classification report of Base-Line, 1 VGG block model, 3 VGG block model, and Ground Truth Model (only VGG block applied) >

To prevent the overfitting problem, the two methods, batch normalization, and L2 regularization were applied to each model. Still, the 1 VGG block model had the biggest number of parameters. Also, both train and validation accuracy and loss graph in the 1 VGG block model showed the most ideal one. However, the overall performance of each train and validation has been degraded after combining with both overfitting prevention methods. The overall accuracy of each model has been decreased. The loss of validation has been vibrated through the epochs of time. The reason why this happened could be discussed.



<Left to Right: loss and accuracy graphs and classification report of Base-Line, 1 VGG block model, 3 VGG block model (overfitting prevention methods applied) >

Discussions

The results from the overfitting prevention combination performance suggested some side effects of Batch Normalization and L2 regularization. Therefore, it should be discussed the reason why they happened and how they can be fixed.

As described in the Batch Normalization section, the technique relies on batch first and second statistical moments (mean and variance) to normalize the hidden layers activations. Therefore, the output values are surely tied to the current batch statistics. Such transformation can add some noise, depending on the kinds of input used in the current batch. Although the regularization was generated from the solution to avoid overfitting problems, relying on batch normalization can make the development process much more difficult than needed. We should always consider the orthogonality matters.

The theoretical access to the L2 regularization suggests why there was difficulty in using the L2 regularization. The lambda, the penalty parameter has the key. If the lambda is at a good value, the regularization helps to avoid overfitting. However, choosing the lambda may be hard. Therefore, we should often use the cross-validation technique. If there are irrelevant features in the input, which do not give any effect to the output, the L2 regularization will give them very small but not zero weights. However, the irrelevant input ideally should have exactly zero weights.

Summary and Conclusions

To summarize this project, the dataset consisted of two classes; normally laser-welded hairpin and abnormally laser-welded hairpin. For further study, a new dataset that contains uniformly distributed abnormal cases should be achieved. After data generation with the loading process, the normalization and the augmentation techniques using the Keras library were applied. Based on the Base-Line CNN architecture, the transformed versions of VGG block architectures were made. To prevent the overfitting problem, the Batch Normalization and L2 Regularization methods were applied. Comparing to the prior Base-Line model performance, the results from the VGG block models showed better accuracy and low loss through the epochs time. However, comparing to the prior step, the result from the combination with the overfitting prevention showed degraded performance. The cross-validation technique to find the best value of lambda would be the best solution for this situation. This helps not only finding the parameter but also evaluating the models on the limited data sample. By the parameter tuning process, not only a batch size but also the epoch time and the learning rate for the optimizer have to be considered. It has been known the smaller learning rates require more training epochs because the little changes lead to the few weights update. Also, the larger learning rates make rapid changes and require fewer training epochs. Additionally, the number of batch normalization layers and the drop-out layer should be reconsidered. Not inserting them into all the gaps between the conv layer and the pooling layer, they should be positioned at the last step of the model architecture.

References and Supplementary materials

[1] Mayr A, Lutz B, Weigelt M, Glabel T, Kibkalt D, Masuch M et a, “Evaluation of Machine Learning for Quality Monitoring of Laser Welding Using the Example of the Contacting of Hairpin Windings”.
[2] J. Vater, P. Schamberger, A. Knoll, and D. Winkle, “Fault Classification and Correction based on Convolutional Neural Networks exemplified by laser welding of hairpin windings”.
[3] X.-D. Gao, Q. Wen and S. Katayama, “Analysis of high-power disk laser welding stability based on classification of plume and spatter characteristics”.
[4] S. Lee, S. Ahn and C. Park, “Analysis of Acoustic Emission Signals During Laser Spot Welding of SS304 Stainless Steel”.
[5] Z. Xiao-Guang, X. Jian-Jian and G. Guang-Ying, “Defects recognition on X-ray images for weld inspection using SVM”.
[6] I. Valanvanis and D. Kosmopoulos, “Multiclass defect detection and classification in weld radiographic images using geometric and texture features”.
[7] A. Khumaidi, E.M. Yuniarno and M.H. Purnomo, “Welding defect classification based on convolution neural network (CNN) and Gaussian kernel”.

* Working Principle of Three-phase Induction Motor | Electrical Article
* Image Classification in Python with Keras | Image Classification (analyticsvidhya.com)
* How to Develop VGG, Inception and ResNet Modules from Scratch in Keras (machinelearningmastery.com)
* Increase the Accuracy of Your CNN by Following These 5 Tips I Learned From the Kaggle Community | by Patrick Kalkman | Towards Data Science
* How to Accelerate Learning of Deep Neural Networks With Batch Normalization (machinelearningmastery.com)
* https://towardsdatascience.com/batch-normalization-in-3-levels-of-understanding-14c2da90a338
* https://www.cs.mcgill.ca/~dprecup/courses/ML/Lectures/ml-lecture02.pdf
* https://towardsdatascience.com/intuitions-on-l1-and-l2-regularisation-235f2db4c261
* https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/
* https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/
* https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53