Hierarchical Convolutional Neural Networks using CCP-3 Block Architecture for Apparel Image Classification

Natthamon Chamnong[1], Jeeraporn Werapun[2], Anantaporn Hanskunatai[3]
Data Science and Computational Intelligence Lab-Department of Computer Science-School of Science, King Mongkut's Institute of Technology Ladkrabang, Bangkok, 10520, Thailand[1, 2, 3]

***Abstract*—In fashion applications, deep learning has been applied automatically to recognize and classify the apparel images under the massive visual data, emerged on social networks. To classify the apparel correctly and quickly is challenging due to a variety of apparel features and complexity of the classification. Recently, the hierarchical convolutional neural networks (H–CNN) with the VGGNet architecture was proposed to classify the fashion-MNIST datasets. However, the VGGNet (many layers) required many filters (in the convolution layer) and many neurons (in the fully connected layer), leading to computational complexity and long training-time. Therefore, this paper proposes to classify the apparel images by the H–CNN in cooperated with the new shallow-layer CCP-3-Block architecture, where each building block consists of two convolutional layers (CC) and one pooling layer (P). In the CCP-3-Block, the number of layers can be reduced (in the network), the number of filters (in the convolution layer), and the number of neurons (in the fully connected layer), while adding a new connection between the convolution layer and the pooling layer plus a batch-normalization technique before passing the activation so that networks can learn independently and train quickly. Moreover, dropout techniques were utilized in the feature mapping and fully connected to reduce overfitting, and the optimizer adaptive moment estimation was utilized to solve the decaying of gradients, which can improve the network-performance. The experimental results showed that the improved H–CNN model with our CCP-3-Block outperformed the recent H–CNN model with the VGGNet in terms of decreased loss, increased accuracy, and faster training.**

Keywords*—Convolutional neural networks (CNN), hierarchical CNN (H-CNN), CCP-3 block (two convolutional layers (CC) and one pooling layer (P) per block), apparel image classification, fashion applications

I. INTRODUCTION

In the Big-data era, social media platforms generate a tremendous volume of image data. There have been initiatives to utilize the valuable image data in a variety of industries, including the business and medical sectors. Due to a vast amount of accessible image data for training and the state-of-the-art technology that provides superior processing capability via the GPU, the unstructured visual data can now be implemented in statistical and data mining applications. Under the GPU technology, it is really simple and fast to analyze the image data. In a previous study, the image data were analyzed using traditional machine learning and image processing techniques [1]. However, typical machine learning and image processing approaches are still limited in their processing capabilities when working with large image data. To overcome the processing restrictions associated with big picture data analysis, deep learning techniques such as Deep Neural Networks (DNN) are applied in the form of Convolutional Neural Networks (CNN) [2]. Currently, a deep learning model, when applied to the image data, provides a CNN architecture that performs well in classifying the image data.

Because apparel products in fashion applications are diverse and difficult to describe, the automatic CNN is frequently used to classify the apparel image data. A fashion-classification system uses a hierarchical structure that can be divided from the coarse to fine hierarchies. Each item in the fine hierarchy can be defined as a higher-level item, such as a t-shirt pullover and a shirt. These three different types of shirts are classified separately but can be combined in the same coarse layered Tops category because of their similarity. However, the classification of features for each hierarchy of items lacks the specific classification criteria and instead is classified based on similar features [3, 4]. As a result, the better categorizing the apparel products by using the CNN architecture is challenging. Applying the efficient CNN method of image classification, which has the advantage of assisting in filtering, categorizing, and product inspection, helps the apparel industry reduce the cost and time, while improving business efficiency [3, 5]. While CNN approaches are popular to the categorization of apparel image data, their tradeoff results (in terms of accuracy and speed) have been questioned. Therefore, many attempts have been made to develop more efficient strategies for optimizing the CNN models for the apparel classification. To improve the classification accuracy [6], a hierarchical classification strategy was used to classify the apparel image.

Recently, the fashion images were classified by using a hierarchical classification system [7]. In a hierarchical structure of fashion types, the Hierarchical Convolutional Neural Networks (H-CNN) was proposed and focused on the VGGNet architecture. The H-CNN was applied to the "Fashion-MNIST" dataset, an improved public image dataset for direct analysis. It is a 28 x 28 grayscale image of 10 classes comprised of 60,000 training photos and 10,000 test images, separated into 3 levels of coarseness: coarse 1, coarse 2, and fine. However, the existing H-CNN and the VGGNet were computed sequentially in deep architectures, where the VGGNet (many layers)

required many filters (in the convolution layer) and many neurons (in the fully connected layer), leading to computational complexity and long training-time. On the other hand, the H-CNN has not yet been implemented to improve the performance in shallow architectures.

Therefore, this study proposes to use the H-CNN in conjunction with our new CCP-3 block architecture, a minimalistic size founded on the concept of a shallow architecture (instead of a deep architecture) to achieve the better performance for not only the accuracy but also the computing time. Based on the popular models from the LeNet and AlexNet designs, the new CCP-3 block was introduced by reducing the number of layers in the network, the number of filters in the convolution layer, and the number of neurons in the fully connected layer, along with a new connection between the convolution layer and the pooling layer. In addition, a batch normalization technique was employed before passing the activation function so that networks can learn independently and train quickly. Moreover, dropout techniques were utilized in the feature map and fully connected to reduce overfitting, and the optimizer adaptive moment estimation was utilized to solve the decaying of gradients. In this study, the hypothesis is that "the integration of selected appropriate architectures in the H-CNN can improve the network performance". In the performance evaluation, The CCP-3 Block architecture has been implemented in the H-CNN model to observe the improvement of the classification accuracy for the apparel image data and observe the speedup of the training time (on the GPU machine) in an experiment. Performance results showed that the CCP-3 Block architecture in the H-CNN model decreases training time significantly and improves the classification accuracy for apparel picture data over the recent VGGNet architecture in the existing H-CNN.

In summary, the main contributions of this study are as follows:

- This study proposes a novel CCP-3 Block architecture to optimize the H-CNN model for the efficient classification of the apparel images.
- This study compares the performance of the H-CNN models based on the existing VGGNet architecture and new CCP- 3Block architecture.

The remainder sections of this paper are organized as follows. Section II summarizes the CNN architectures and the related works. Section III presents the proposed CCP-3 Block architecture. Section IV illustrates the experiment on the fashion-MINIST dataset. Section V presents the experimental results and Section VI discusses the conclusion of this study and the future study.

II. Related Works

In this section, an overview of the convolutional neural networks (CNN), the modern CNN architectures, and the optimization techniques of CNN are reviewed and a related work, called the hierarchical CNN (H-CNN) using VGG16 and VGG19 architectures, was presented to classify the apparel images.

A. Convolutional Neural Network (CNN)

A convolutional neural network (CNN) is a neural network model of the human-vision emulation that perceives a space as sub-sectors and integrates the sub-sectors together to identify "what is visible". Human perceptions of sub-areas are shaped by sub-area features, such as lines and color contrasts. Humans recognize that "the focused area is defined by a straight line or a contrasting color" because they combine both of the interested area and the surrounding area concurrently [8, 9]. The construction of CNN is divided into two main components [10-12]: 1. the first one is the feature extraction layer (for extracting features) and 2. the subsequent section is the classification layer (to educate and classify), which will ensure that the connection layer is fully connected. In the feature extraction layer, there are three sub-layers: Convolution Layer, ReLU (Rectified Linear Units) Layer, and Pooling Layer. In the classification layer, there is only one fully connected layer, which resembles a node in the neural network. Each of those layers has the particular and different functions.

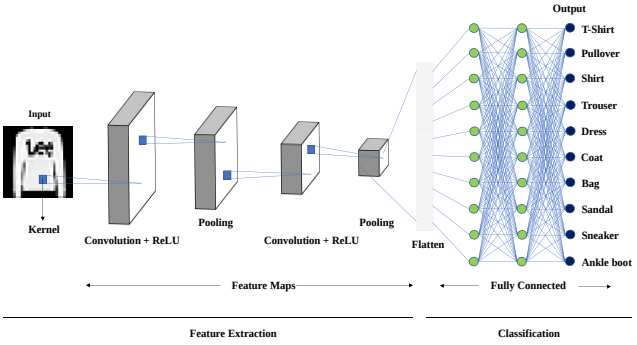


Fig. 1. Structure of CNN.

Fig. 1 describes the standard structure of the CNN, which consists of the following layers:

1) Input layer: Read the input data of the image and pass it to the neural network.

2) Convolutional layer: Create a sliding window (filter or kernel) that scans the input image to make a feature map. Initially, it scans the image to extract image elements such as borders, colors, and shapes, where the working principle starts with the convolution of the existing input image with the kernel and shifting it to the position of the next kernel. By scrolling the kernel position, the scroll distance can be adjusted. Repeat the same process, until all points of the input image are concerned. The convolution using the formula given below.

$$S_{ij} = (I * k)_{ij} \tag{1}$$

$$(I * k)_{ij} = \sum_{a=0}^{m-1} \sum_{b=0}^{n-1} I_{i+a,j+b}\, k_{a,b} \tag{2}$$

S refers to the result of convolution at any position. I refers to image input. k refers to kernel. i, j refers to any position. m, n refers to the number of rows and columns.

3) Rectified linear unit (ReLU): Perform a nonlinear activation function. The function given below

$$f(x) = \max(0, x) \quad (3)$$

4) Pooling layer: After the Convolutional layer(s) in the structure of a CNN, a Pooling layer is inserted. It calculates the maximum or average of the input and reduces the output of the Convolutional layers by sliding the filter with a specific shape and stride size.

5) Fully connected layer: Configure the output and display in the form of a multiclass logistic classifier.

6) Output Layer: Display the results of the classification. However, CNNs can have different layer elements in different architectures because each CNN consists of a layer convolutional for creating feature maps and pooling for the dimensionality of feature maps. By stacking these layers [7], we can formulate various CNN architectures.

B. Architectures of Convolutional Neural Networks

1) *LeNet architecture*: The study to optimize the CNN model with a very well-structured architecture is another possibility to increase the performance of the CNN model. LeCun et al., [13] developed LeNet-5 in 1998, a network based on the CNN concept. In the convolutional layer, there are seven classification levels for numbers. Numerous banks employ it to identify the handwritten digits on digital checks using a 32x32 pixel image. Increasing the processing capability of higher resolution images requires a larger neural network layer and many layers. The LeNet-5 architecture is composed of two convolutional layers, two pooling layers, and three fully connected layers.

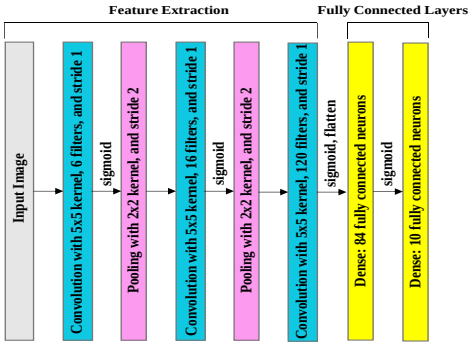


Fig. 2. Architecture of LeNet-5.

Fig. 2 describes the structure of LeNet-5 architecture. There are three convolution layers within the architecture, with two pooling and two fully connected. In each of the three convolution layers, the kernel size is 5x5 and the number of strides is 1. The distinction lies in the number of filters, with the first layer, second, and third having 6, 16, and 120 filters, accordingly. In the pulling layer, the kernel size is 2x2 and the number of strides is 2, which is identical to both layers. In a fully connected layer, the number of neurons in the first is 84, whereas the number of neurons in the second is dependent on the number of outputs. Sigmoid will be used as the activation function.

2) *AlexNet architecture*: AlexNet is a neural network, developed in 2012 by Krizhevsky et al., [14] which was intended to classify 1.2 million high-resolution images with dimensions of 224x224x3, with images classified into 22,000 different classes. AlexNet achieves a top-5 test error rate of 16.4% in the ImageNet LSVRC-2012 contest. The AlexNet architecture is composed of 5 convolutional layers, 3 pooling layers, and 3 fully connected layers. In addition, it uses Rectified Linear Unit (ReLU) for the nonlinearity function, which is faster than Hyperbolic Tangent (tanh) function.

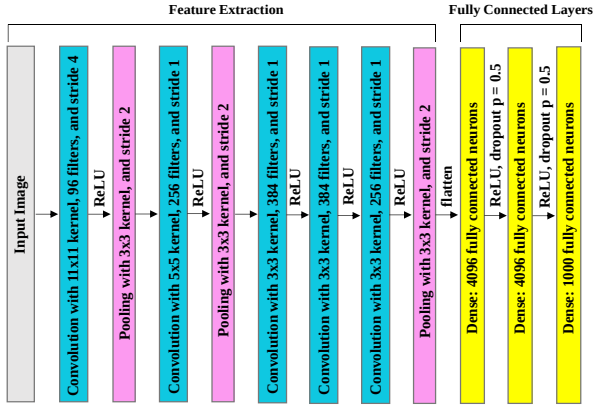


Fig. 3. Architecture of AlexNet.

Fig. 3 describes the structure of AlexNet architecture. There are five convolution layers in the architecture, with three pooling and three fully connected. The kernel sizes for the first and second convolution layers are 11x11 and 5x5, while the kernel sizes for the third, fourth, and fifth layers are all 3x3. In five convolution layers, there are 96, 256, 384, 384, and 256 filters, respectively, with the first layer the number of strides is 1, and in the remaining four layers, the strides are 4. In the pulling layer, the kernel size is 3x3 and the number of strides is 3, which is identical to all layers. In a fully connected layer, the number of neurons in the first and second is 4096, and the third is dependent on the number of outputs. However, in this architecture, the dropout rate is 0.5 and the activation function is used as ReLU.

3) *VGGNet architecture*: VGGNet was invented by the Visual Geometry Group as an architecture standard of deep convolutional neural network (deep CNN) with multiple layers. The most popular depth of the VGGNet architecture is VGG16 and VGG19 because the VGG16 and VGG19 architectures are the basis of ground-breaking object recognition models. The VGGNet architecture, developed as a deep neural network to surpass baselines on many tasks and datasets beyond ImageNet, consists of 16 and 19 layers of convolutional and fully connected layers. In competitive LSVRC-2014, the VGGNet won the 1st runner-up with less than 10% error rate and deeper layers containing 16 convolutional and fully connected layers. It uses 3 ×3 sized filters, a stride of 1 and 2 ×2 sized pooling, and a stride of 2 from the beginning to the end of the network. It also uses ReLU for nonlinearity function and is trained by batch stochastic gradient descent [15]. The structures of the VGG16 and VGG19 architectures are shown in Fig. 4 and Fig. 5.

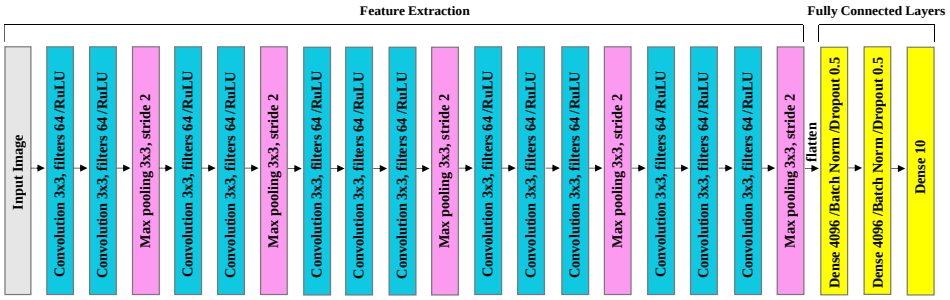


Fig. 4. Architecture of VGG16.

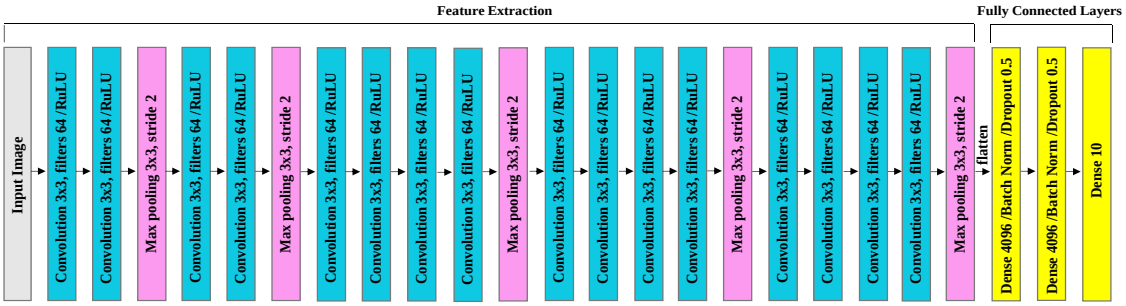


Fig. 5. Architecture of VGG19.

The number 16 and 19 in the name VGG (Visual Geometry Group) refer to the depth of 16 and 19 layers in the deep CNN. This means that VGG16 and VGG19 are extensive networks, where each of them has a total of around 138 million parameters. While VGGNet is popular in the modern standard, it is a huge network. However, the simplicity of the VGGNet architecture makes this network being more appealing. For example, there are a few convolution layers followed by a pooling layer that reduces the height as well as the width. When considering the number of filters, 64 filters are available and can be double to 128 filters and 256 filters. Finally in the last layer, we can use 512 filters.

In summary, the major differences of three architectures (LeNet, AlexNet, VGGNet) are focused on the architecture size and the activation function. In the initial periods of CNNs, the CNN architecture was a small structure with limited computational resources. Later, the larger CNN architectures have been constructed in response to the development of computational resources to be able to support the larger architecture designs. However, in this era the development of many CNN designs aims to decrease loss and increase accuracy, while being able to train models in fast or efficient time.

C. Guide to Improving CNN

1) Optimizer: Optimizers can be explained as a mathematical function to modify the weights of the network, according to the gradients and additional information, which depend on the formulation of the optimizer. The optimizers are built upon the idea of gradient descent, the greedy approach of iteratively decreasing the loss function by following the gradient. However, different optimizers will affect the model sensitivity and learning accuracy [16-19]. As a result, it is essential to use an appropriate optimizer for data and developed models.

2) Regularization: Regularization is the process of learning from the training datasets and modifying the model to be more efficient at predicting and reducing loss from the unseen data. The regularization is used to solve the issues of underfitting or overfitting. To address the underfitting problem of the neural network model, usually the number of layers and nodes in each layer can be increased but this can cause the overfitting [20-22]. Therefore, the regularization is a frequently mentioned solution, which is very simple to be implemented. The regularization technique consists of augmentation, batch normalization, and dropout, where their functions are defined as follows:

a) Augmentation: Augmentation is a technique to increasing the amount of data to train by generating the more data. In the case of image data, increasing a variety of images includes rotating images, zooming images, shifting images horizontally, shifting images vertically, and shear images.

b) Batch normalization: Batch Normalization is a technique for scaling the data to adjust their values to the specified limits before exporting from the node to the next layer input. For example, a feature engineering procedure converts the grayscale image from 0-255 to 0-1 by dividing the original color value by 255. For data normalization, several well-known methods can be utilized, such as min-max normalization or standardization.

c) Dropout: Dropout is an effective process of regularizing neural networks to avoid the overfitting. During training, the dropout layer cripples the neural network by removing the hidden units stochastically.

3) Efficient shallow learning as an alternative to deep learning: In 2022, Y. Meir et al. [23] discusses the realization of complex classification tasks using deep learning architectures with many convolutional and fully connected hidden layers. The authors demonstrate that with a fixed ratio between the depths of the first and second convolutional layers, the error rates of shallow architectures like the LeNet and VGG-16 can decay as a power law with the number of filters in the first convolutional layer. This phenomenon suggests a quantitative hierarchical time-space complexity among machine learning architectures and calls for further examination using various databases and architectures. The conservation law along the convolutional layers is found to minimize error rates. The study emphasizes the efficient shallow learning and its potential for implementation using dedicated hardware developments.

D. Hierarchical Classification

Hierarchical classification is a system of grouping things (or objects) according to a hierarchy, such as levels and orders. A hierarchical classifier classifies the input data according to the output categories, which are defined subsumptively. Classification begins at a basic level with the fine-detailed input data. The classifications of the separate bits of the image data are then integrated and elevated to a higher level iteratively until a single or defined output is obtained. This final output represents the overall result of the data classification.

In 2015, Yan et al. [24] proposed the first trial of hierarchical image classification using a deep learning approach. To resolve class confusion in the proposed model, Hierarchical Deep Convolutional Neural Networks (HD-CNN) employed an initial coarse classifier CNN to differentiate easily separable classes (or coarse classes) from fine classes. Additionally, the HD-CNN model could be implemented without increasing the training complexity. However, that model encountered some limitations, which were that it required two steps of training. The first step was to train the coarse and fine categories and the second step was to fine-tune the coarse and fine categories. Moreover, the HD-CNN model could not be used to classify many levels of hierarchy since it included one coarse category and one fine category only for an overall of two levels.

Later in 2017, the Branch Convolutional Neural Network (B-CNN) was proposed by Zhu and Bain [25] to solve the limitation of HD CNN.

Due to previous CNN research during 2015 - 2019, the hierarchical CNN study along with the particular application could improve the accuracy in the experiment. Therefore, implementing the hierarchical classification to optimize the CNN models to respect the diversity of datasets, applications, and CNN architectures is interesting.

In 2019, Seo and Shin [7] introduced the Hierarchical Convolutional Neural Networks (H-CNN) for the categorization of fashion images in the Fashion MNIST image data, where the fashion imagery obtained from Zalando is similar to the MNIST Dataset's handwritten numeric dataset, a refined fashion image. That study employed the large-scale VGGNet neural networks as an experimental model. In performance evaluation (on the Fashion-MNIST dataset), accuracy results of the usage of H-CNN under the VGGNet architecture outperformed those of the simple VGGNet network.

In 2021, Q. Zhu et al. [26] discusses the use of drone imagery in automated inspection for surface defects in infrastructure. The proposed approach in the paper is a deep learning method that uses hierarchical convolutional neural networks with feature preservation (HCNNFP) and an intercontrast iterative thresholding algorithm for image binarization. The technique is applied to identify surface cracks on roads, bridges, or pavements, and is compared with existing methods on various datasets using evaluation criteria including the average F-measure. The proposed technique outperforms existing methods on various tested datasets, especially for the GAPs dataset, demonstrating the merits of the proposed HCNNFP architecture for surface defect inspection.

This study is interested in developing the H-CNN (Hierarchical CNN) under the more efficient architectures for the fashion applications (in Section III). Therefore, the previous study [7] is the main related work, see detail in Section II E.

E. Original Hierarchical Convolutional Neural Network (H – CNN) Model

With regards to the original H-CNN model under VGG16 and VGG19 architectures [7], both VGG16 and VGG19 are composed of five building blocks as shown in Fig. 6 and Fig. 7.

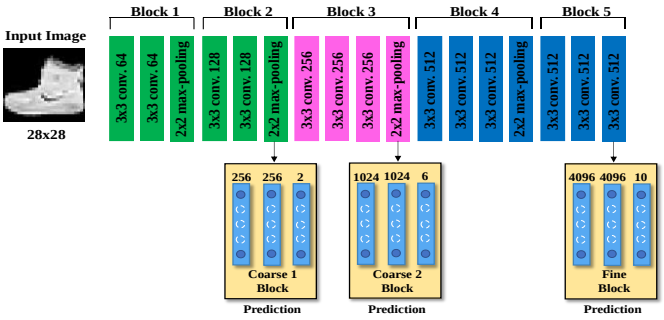


Fig. 6. Architecture of VGG16 H–CNN model.

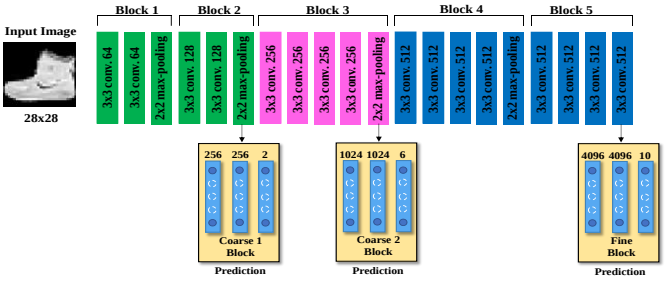


Fig. 7. Architecture of VGG19 H–CNN model.

In the VGG16 H–CNN model, the first and second building blocks consist of two convolutional layers and 1 pooling layer, the third and fourth blocks consist of 3 convolutional layers and 1 pooling layer, and the fifth building block has 3 convolutional layers.

In the VGG19 H–CNN model, the first and second building blocks have 2 convolutional layers and 1 pooling layer, the third and fourth blocks have 4 convolutional layers and 1 pooling layer, and the fifth building block consists of 4 convolutional layers.

The filter size and number of filters in the convolution layer are the same for both VGGNet architectures, with the filter size being 3x3 throughout the model. For the number of filters, they can be divided as follows: In the first block, there are 64 filters, the second block 128 filters, the third block 256 filters, and the fourth and fifth blocks 512 filters.

Moreover, these H–CNN models also use ReLU for activation function, batch normalization for initialization, and dropout for regularization. In the final block denoted as the fine prediction block, the softmax function is used to classify 10 fine classes.

However, this model has three additional blocks below followed by a prediction block. In each block, there are labels for 3 levels of classification, which makes it different from the basic model. The first block is for course-level, the second block is for course-level, and the last block is for fine-level. All three additional blocks are composed of fully connected neural networks. As the input image goes through the H–CNN model, three prediction values of coarse 1 level, coarse 2 level, and fine level will be computed in order. For example, when an input image of a sweater is inserted, the first coarse level block will indicate 'clothes', the second coarse level block will indicate 'tops', and the final block will indicate 'pullover' as output predictions.

III. Proposed Method

Applying the convolutional neural network (CNN), especially the efficient deep learning, to fashion applications (to achieve not only the high accuracy but also the fast training) is challenging under the massive visual data emerged on the current social networks. Recently (2019), the hierarchical CNN (H–CNN) was proposed to classify the fashion-MNIST datasets with a capability of high accuracy. However, in that H-CNN the applied VGGNet (the deep architecture) is composed of many layers, many filters (in the convolution layer), and many neurons (in the fully connected layer), leading to computational complexity and long training-time.

According to the hypothesis (in the fashion classification) believe that the shallow architecture plus a few proper functions can yield good results as the deep architecture, while can take faster training-time to solve computational complexity problems. In benefit summary of existing architectures, the (deep) VGGNet architecture requires many layers, many filters, and many neurons with long training-time for high accuracy, while the (shallow) AlexNet architecture require less training time (with shallow layers) but less accuracy. Therefore, we focus on studying the novelty and strength of the architecture for the H-CNN model to decrease loss, increase accuracy, and fast training-time.

This study proposes to classify the apparel images with the H–CNN model using the new CCP-3 Block architecture to retain the accuracy as the VGGNet architecture within the less training-time as the AlexNet architecture, where each building block consists of double convolutional layers (CC) and one pooling layer (P). As mentioned earlier, our proposed CCP-3 Block architecture was inspired by the fast LeNet and AlexNet microarchitectures (with shallow layers).

In Section III A, the new CCP-3 Block architecture is proposed first for classifying the apparel/fashion image. In Section III B, the H–CNN model using the CCP-3 Block architecture is presented for the completed classification. In Section 4, the experiment is conducted on the fashion-MNIST datasets to compare the performance of CCP-3 Block architecture. Finally, the experimental results are presented in Section V.

A. CCP-3 Block Architecture

The CCP-3 Block architecture is a shallow-layer architecture, see details in Fig. 8, which can reduce the number of layers (in the network), the number of filters (in the convolution layer), and the number of neurons (in the fully connected layer) of the deep-layer architecture, while adding a new connection between the convolution layer and the pooling layer.

The CCP-3 Block architecture has only three blocks shown in Fig. 9. Each building block consists of two convolutional layers and one pooling layer. The 3×3 sized filters with a stride of 1 are used in all convolutional layers. In the first building block, 64 filters are concatenated and in the second block, first convolution has 128 filters, second convolution has 256 filters and 512 filters in the third block. For the pooling layers, the 2×2 size max-pooling is done with a stride of 2. In a fully connected layer, there are 3 layers, where in the first and second layers we define the number of neurons as 1024 neurons, and in the last layer, we define 10 neurons into 10 classes using the softmax function.

Moreover, ReLU was used for the activation function, batch normalization for initialization, and dropout for regularization. In the structure of CCP-3 Block architecture, batch normalization will be implemented in order to ensure that for any parameter value after the convolution layer, the network always produces activations with the desired distribution. So, the batch normalization layer is inserted right after the convolution layer, but before feeding into ReLu activation [27]. To reduce overfitting, dropout was added into both building blocks and the fully connected layer. In the building block, dropout was defined between the convolution layer and after the pooling layer. The fully connected, dropout was defined after batch normalization layer. Throughout the architecture, we set the dropout value to 0.3.

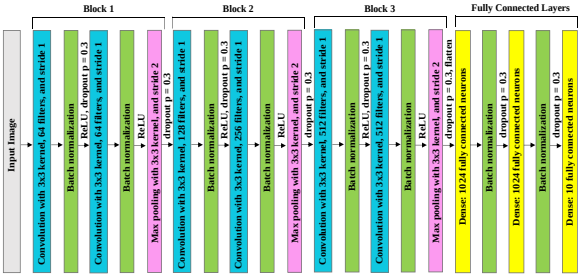


Fig. 8. Details of CCP-3 block architecture.

B. H-CNN using CCP-3 Block Architecture

The H-CNN model was implemented in conjunction with CCP-3 Block architecture by adding additional blocks below each main block, followed by a prediction block, shown in Fig. 9. The additional blocks have the same functions and properties as those blocks in the original H-CNN. Each additional block contains labels indicating one of three classification levels. The first block is intended for course-level instruction, the second block is intended for course-level instruction, and the final block is intended for fine-level instruction. Each of these three blocks is composed entirely of fully connected neural networks. As the input image passes through the H–CNN model, three prediction values will be computed in order: coarse 1 level, coarse 2 levels, and fine level.

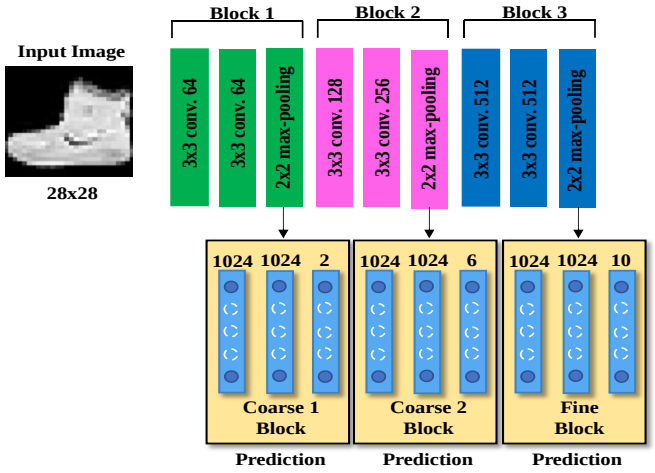


Fig. 9. Architecture of H–CNN CCP-3 block model.

IV. Experiments

To evaluate the performance of CCP-3 block architecture, the H-CNN model was implemented in incorporated with CCP-3 Block architecture. The experimental results were compared to those of the original H-CNN model using VGG16 and VGG19 architectures on the same environment. See the improved results in Section V in terms of increased accuracy and decreased computing-time.

A. Environment Setup

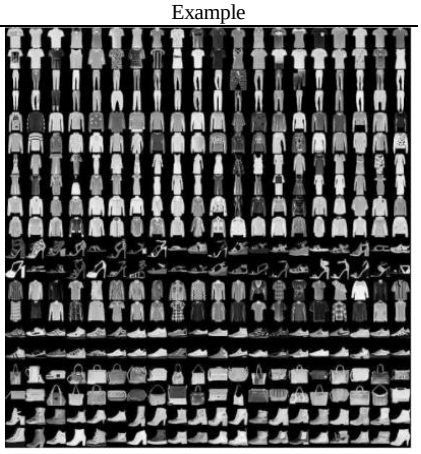
This experiment implemented and operated the above 3-model programs on the google colaboratory. This programming environment investigated a GPU runtime (speed up execution), the GPU machine used in this operation is the Tesla P100-PCIe.

B. Dataset

This paper uses Fashion MNIST image dataset (see Table I). This fashion image dataset is collected from Zalando, which is similar to the MNIST dataset handwritten digit classification. In this standard dataset, each grayscale image is a square size of 28 ×28 pixels and all images are divided into 10 classes: t-shirt, trouser, pullover, dress, coat, sandal, shirt, sneaker, bag, and ankle boot. Each class contains an equal number of samples. The 60,000 samples are used for training and the 10,000 samples are used for testing. In the hierarchical structure, these 10 classes can be restructured into two coarse classes and one fine class as shown in Fig. 10.

Each first-level coarse class consists of the second-level coarse classes and each second-level class consists of the fine-level classes. The first-level coarse class consists of 'clothes' and 'goods'. In the second-level coarse class, the 'clothes' contain 'tops', 'bottoms', 'dresses', and 'outers' as well as the 'goods' contain 'accessories' and 'shoes'. Below the second-level coarse classes, there are fine-level classes consisting of 't-shirt', 'pullover', and 'shirt' in 'tops', 'trouser' in 'bottoms', 'dress' in 'dresses', 'coat' in ‘outers', 'bag' in 'accessories'; 'sandals', 'sneaker', and 'ankle boots' in 'shoes'. For hierarchical matching in the H-CNN models, the first-level coarse classes are represented by green, second-level classes are represented by pink, and fine-level classes are represented by blue [7]. Each color in Fig. 10 matches the original H-CNN models in Fig. 6 and Fig. 7 and the H-CNN using CCP-3 Block architecture in Fig. 9.

TABLE I. Fashion-Mnist Dataset

Label	Description	Example
0	T-Shirt/Top	
1	Trouser	
2	Pullover	
3	Dress	
4	Coat	
5	Sandals	
6	Shirt	
7	Sneaker	
8	Bag	
9	Ankle boots	

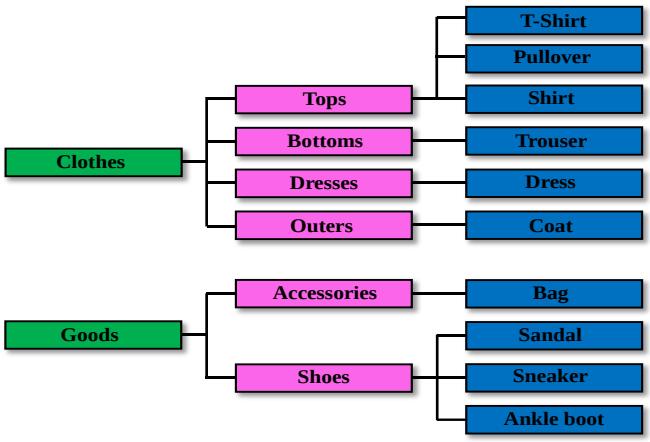


Fig. 10. Hierarchical classes of dataset of the original H-CNN.

C. Parameter Setting

1) Parameter setting of the original H-CNN using VGG16 and VGG19 [7, 16, 28, 29]: To train two original H-CNN models, the parameters were set as follows: A number of epochs were set to 60 times and the size of the batch was set to

128. There were variations in learning rate as 0.001 used in the initial stages, 0.0002 after the 42th epoch, and 0.00005 after the 52th epoch. Stochastic gradient descent was applied by using 0.9 of momentum. To reflect differences in the importance of each level of the class the loss weight values were added when training models. The changes in loss weights were set to [0.98, 0.01, 0.01] in the first epoch, [0.10, 0.80, 0.10] in the 15th epoch, [0.1, 0.2, 0.7] in the 25th epoch, [0, 0, 1] in the 35th epoch.

2) Parameter setting of the H-CNN using CCP-3-Block: In this study, we set similar parameters as the original H-CNN models, such as a number of epochs, learning rates, the changes in loss weights. However, the stochastic gradient descent is not used in the model because the architecture of the CCP-3 block is small. In addition, an appropriate optimizer was used to reduce overall losses and improve accuracy [30, 31]. Adaptive moment estimation is applied by using 0.9 of beta1, 0.999 of beta2, and 1e-07 of epsilon.

V. Results

In order to evaluate the performance of the H-CNN model using CCP-3 Block architecture, the performance was compared of the H-CNN CCP-3 Block model to the original H-CNN models using VGG16 and VGG19 architectures. Table II shows the results (the final loss, accuracy of the test, and the training time) of each model. The CCP-3 Block architecture has a loss of 0.2714, while the VGG16 and VGG19 architectures have the loss of 0.3781 and 0.3863. About the accuracy of 0.9490, while the others have the accuracy of 0.9352 and 0.9341. The CCP-3 Block model has the fastest training time of 18.21 minutes, while the others have 20.33 and 27.28 minutes (H-CNN CCP-3 Block is 10.32 percent faster than H-CNN VGG16 and 32.87 percent faster than H-CNN VGG19.). In comparison, the CCP-3 Block model has lower loss, greater accuracy, and less training time than the other two models.

Table III shows the test accuracy results (0.8970-0.9410) of previous researches (i.e., data mining methods and other CNN models), compared to the test accuracy (0.9490) of the CCP-3 Block architecture on the Fashion MNIST dataset. The existing CNN2 and CNN2 + BatchNorm + Skip models were presented by Bhatnagar, Ghosal, and Kolekar (2017), where the CNN model consisting of two convolutional and max-pooling layers (or CNN2), trained by batch normalization (or BatchNorm) with residual skip connections (or skip) to compare the results with those of Support Vector Classifier (SVC) and Evolutionary Deep Learning (EDEN). Later, the accuracy results were improved by the VGG16 and VGG19 based models. Finally, the accuracy result was improved by the CCP-3 Block based model and in this study the CCP-3 Block architecture could generate the best test accuracy when combined with the hierarchical CNN (H-CNN) model.

This study focused to improve the H–CNN model by using the CCP-3 Block architecture over the VGGNet architecture (VGG16 and VGG19). Overall, the loss and accuracy were compared (in training and testing) of each H-CNN model in Table IV. For testing set, the H–CNN using the CCP-3 Block architecture (H-CNN CCP-3 Block) has lower loss (0.2714) than those (0.3781 and 0.3863) of VGG16 and VGG19 and higher accuracy (0.9490) than those (0.9352 and 0.9341) of VGG16 and VGG19. However, when looking at the training set, the H-CNN CCP-3 Block model had a loss of 0.0218 and an accuracy of 0.9920, while the original H-CNN (VGG16, VGG19) models have the better training results because in the H–CNN CCP-3 Block model we added the dropout to both of the building block and in the fully connected layer to solve the overfitting problem, leading to a reliable final-loss and a realistic accuracy (0.9920 < 1.0 (overfitting)) in the training but the better performance in the testing (on the unseen data) with the less final-loss and the higher accuracy.

TABLE II. The Comparison of Final Loss, Accuracy, and Training Time of the Existing H-CNN Models (H–CNN VGG16, H–CNN VGG19) and Our H–CNN CCP-3 Block Model

Model	Test		Training Time (minutes)
	Loss	Accuracy	
H-CNN VGG16	0.3781	0.9352	20.33
H-CNN VGG19	0.3863	0.9341	27.28
H-CNN CCP-3-Block	**0.2714**	**0.9490**	**18.21**

TABLE III. The Comparison of Classification Results on Fashion MNIST Dataset by Previous and Our Researches

Model	Test accuracy
SVC	0.8970
EDEN	0.9060
CNN2	0.9117
CNN2 + Batch Norm + Skip	0.9254
VGG16 based model	0.9289
VGG19 based model	0.9290
CCP-3 Block based model	0.9410
H-CNN CCP-3 Block model	**0.9490**

TABLE IV. The Train and Test Comparison (in Final Loss and Accuracy) of the Existing H-CNN Models (VGG16 H–CNN, VGG19 H–CNN) and Our CCP-3 Block H–CNN Model

	Train		Test	
	Loss	Accuracy	Loss	Accuracy
H-CNN VGG16	0.0002	1.0000	0.3781	0.9352
H-CNN VGG19	0.0004	1.0000	0.3863	0.9341
H-CNN CCP-3-Block	**0.0218**	**0.9920**	**0.2714**	**0.9490**

Moreover, observe that the H–CNN CCP-3 Block model could converge faster than the H–CNN VGG16 and VGG19 models, as shown in Fig. 11 (H-CNN VGG16), Fig. 13 (H-CNN VGG19), and Fig. 15 (H-CNN CCP-3 Block). The more epochs the less in loss until 60 epochs the losses were stable. Meanwhile, the accuracy value of our H–CNN CCP-3 Block model was greater and converged more quickly than the existing models, as shown in Fig. 12 (H-CNN VGG16), Fig. 14 (H-CNN VGG19), and Fig. 16 (H-CNN CCP-3 Block). In

particular, Table V shows the improved performance (a numbers of specific loss and accuracy values) in each epoch (from epoch 1 to epoch 60) of each model.

In summary, the H–CNN CCP-3 Block model can achieve the better performance than the H-CNN VGG16 and VGG19 models (Table II, Table IV, Table V) and other state-of-the-art models (Table III) to classify images in the Fashion-MNIST dataset based on deep learning architectures. The CCP-3

TABLE V. LOSS AND ACCURACY PER EPOCH OF THREE H-CNN MODELS (VGG16, VGG19, AND OUR CCP-3 BLOCK)

Epoch	H-CNN VGG16 Model				H-CNN VGG19 Model				H-CNN CCP-3-Block Model			
	Train		Test		Train		Test		Train		Test	
	Loss	Accuracy	Loss	Accuracy	Loss	Accuracy	Loss	Accuracy	Loss	Accuracy	Loss	Accuracy
1	2.4639	0.3160	1.0611	0.6521	2.8517	0.2341	1.2562	0.5871	0.6346	0.7744	0.6681	0.7867
5	0.7418	0.7664	0.5173	0.8187	0.7884	0.7494	0.5454	0.8048	0.2772	0.8963	0.2869	0.8949
10	0.5202	0.8335	0.4425	0.8560	0.5629	0.8204	0.4583	0.8445	0.2307	0.9137	0.3253	0.8861
15	0.4298	0.8609	0.4331	0.8630	0.4351	0.8601	0.4154	0.8646	0.1846	0.9311	0.2048	0.9283
20	0.3224	0.8936	0.2938	0.9005	0.3410	0.8875	0.3643	0.8803	0.1572	0.9417	0.1972	0.9315
25	0.1999	0.9315	0.2874	0.9095	0.1978	0.9312	0.2881	0.9049	0.1241	0.9532	0.2135	0.9279
30	0.1332	0.9533	0.2820	0.9142	0.1419	0.9504	0.2725	0.9198	0.0998	0.9625	0.2017	0.9374
35	0.0515	0.9823	0.3646	0.9138	0.0533	0.9811	0.3678	0.9097	0.0763	0.9716	0.2112	0.9370
40	0.0471	0.9837	0.3758	0.9147	0.0480	0.9834	0.3415	0.9201	0.0627	0.9770	0.2309	0.9410
45	0.0044	0.9988	0.3598	0.9313	0.0043	0.9988	0.3353	0.9304	0.0364	0.9865	0.2395	0.9473
50	0.0012	0.9998	0.3494	0.9328	0.0013	0.9997	0.3708	0.9325	0.0293	0.9895	0.2600	0.9459
55	0.0007	1.0000	0.3768	0.9348	0.0005	1.0000	0.3832	0.9338	0.0228	0.9918	0.2689	0.9495
60	0.0002	1.0000	0.3781	0.9352	0.0004	1.0000	0.3863	0.9342	0.0218	0.9920	0.2741	0.9490

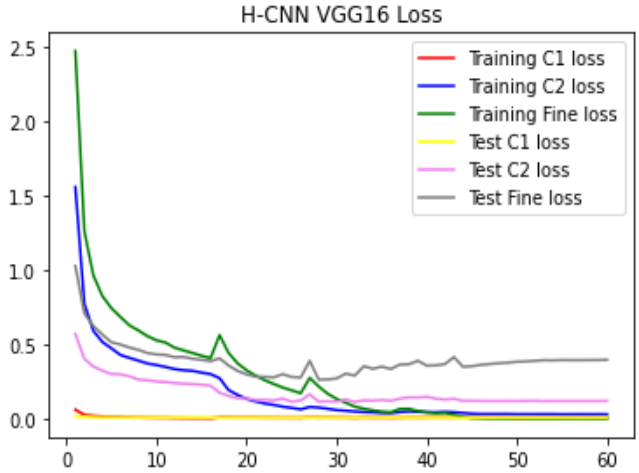


Fig. 11. Loss per epoch in H–CNN VGG16 model.

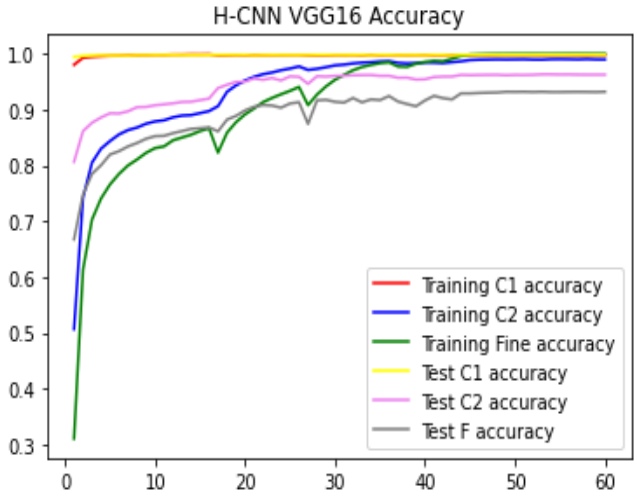


Fig. 12. Accuracy per epoch in H–CNN VGG16 model.

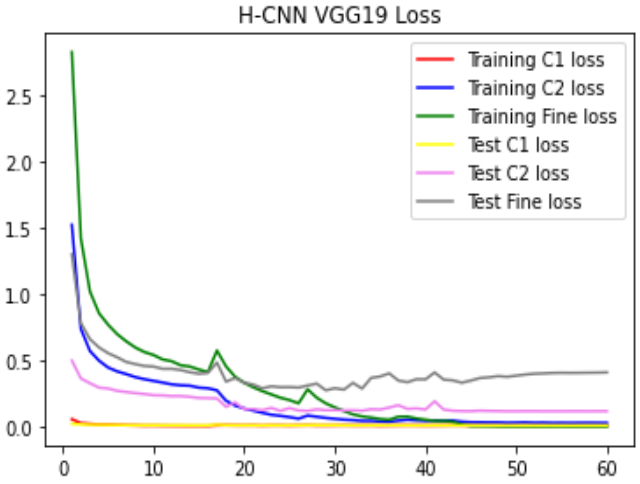


Fig. 13. Loss per epoch in H–CNN VGG19 model.

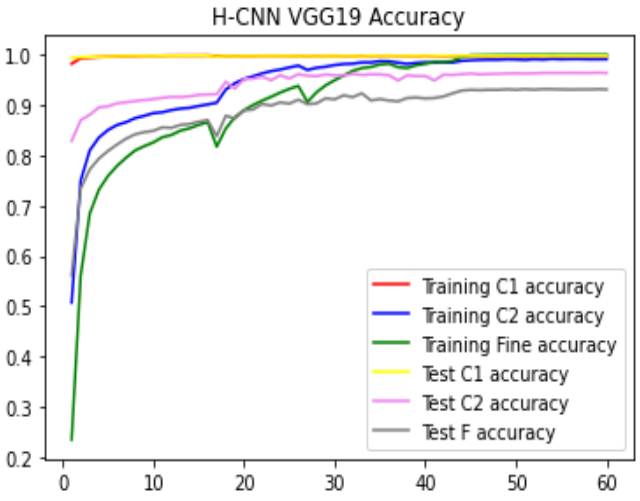


Fig. 14. Accuracy per epoch in H–CNN VGG19 model.

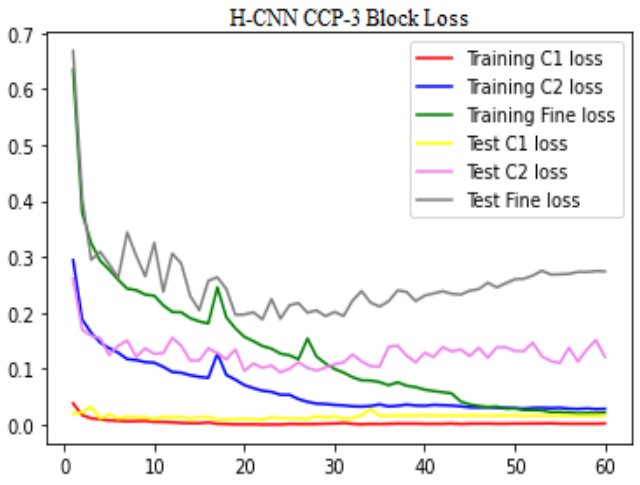


Fig. 15. Loss per epoch in H–CNN CCP-3 Block model.

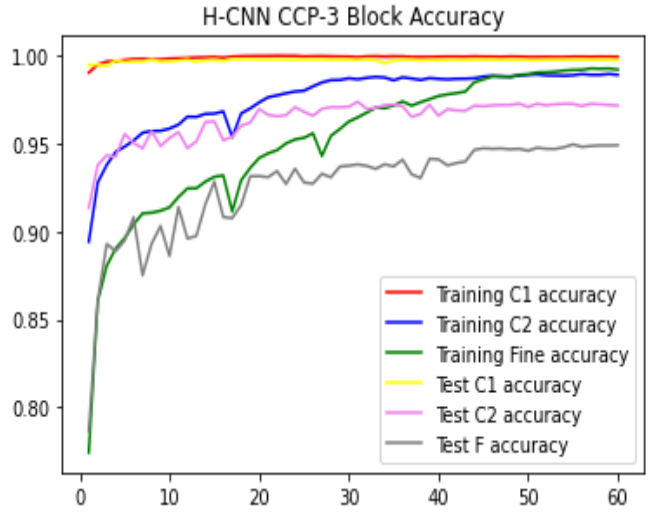


Fig. 16. Accuracy per epoch in H–CNN CCP-3 Block model.

Block design starts with a shallow-layered architecture (combine two convolution layers followed by a pooling layer) and redesign with only three blocks followed by the fully connected layers and add the batch normalization before the activation function so that networks can learn independently and train quickly as well as add the dropout layer in feature extraction and fully connected to reduce the overfitting. Moreover, an optimizer (adaptive moment estimation) was used to solve the decaying of gradients, which can improve the network performance. As a result, the H–CNN CCP-3 Block model has a faster training time and the better performance in testing (decreased loss and increased accuracy). For the image classification in the Fashion-MNIST dataset, the problem of multi-class classification error can be solved by the H–CNN CCP-3 Block model.

VI. Discussion

As presented in the test results section, the H– CNN model uses a shallow layered CCP-3 Block architecture, which provides the best performance in both training speed and classification accuracy. However, when considering the CCP-3 Block architecture used in the classification of apparel images, our design is simplified combine two convolution layers followed by a pooling layer, designed with only three blocks followed by fully connected layers, and adding batch normalization before the activation function. Moreover, adaptive moment estimation is also used to optimize the model (see detail in Section III A). We call this the CCP-3 block base model. Table V shows the test accuracy of the CCP-3 Block base model is 0.9410, which is more accurate than the H-CNN used VGG16 and VGG19 architectures but we perceive something in the table confusion matrix of the CCP-3 Block base model.

Table VI shows the confusion matrix of the CCP-3 block base model. In the case of the shirt category, misclassification samples of 88 T-shirt images, 43 pullover images, and 49 coat images reveal that these three categories locate closer among the 10 categories. The same is the case for the ankle boots category, with misclassification samples of 6 sandal images,

and 34 sneaker images; when these categories are similar. It is reasonable to categorize similar images into a hierarchy.

Observing Table VI, it is possible that the accuracy of the CCP-3 block base model could be increased further if the training images were hierarchically categorized. Therefore, we have applied the Seo and Shin [7] fashion image classification inference to categorize images into a hierarchy as shown in Fig. 10. It is used in conjunction with the CCP-3 Block architecture, which we have designed to support hierarchical classification (see detail in Section III B). The results showed that with the use of hierarchical image classification in combination with the CCP-3 block architecture, accuracy increased to 94.90%. (shown in Table III). When considered in the confusion matrix of the H-CNN CCP-3 Block model (shown in Table VII), in the case of the shirt category, the misclassification was reduced. T-shirt, pullover, and coat were previously misclassified from 88, 43, and 49 images reduced to 66, 33, and 40 images respectively. The same is the case for the ankle boots category, the misclassification is reduced as well. Sandal and sneaker were previously misclassified from 6 and 34 images and reduced to 4 and 26 images respectively. Therefore, categorizing similar images into a hierarchy for classification can increase their accuracy.

The CCP-3 Block base model, a simplified version of the H-CNN model, achieves high accuracy in apparel image classification. However, the model experiences misclassificationions within visually similar categories such as shirts and ankle boots. To address this, we propose a hierarchical classification approach using the Seo and Shin fashion image classification inference. By combining this approach with the CCP-3 Block architecture, the model's accuracy improves significantly to 94.90%. The hierarchical classification effectively reduces misclassifications within similar categories, demonstrating the value of categorizing visually similar images into a hierarchy for improved accuracy. Additionally, pre-defining hierarchical labels of the dataset can also be done by the data-driven method before training the model we want. By considering the classification of the data based on the consideration of the result of the confusion metric.

TABLE VI. CONFUSION MATRIX OF CLASSIFICATION RESULT WITH FASHION MNIST DATASET USING CCP-3 BLOCK BASE MODEL

		Predict label									
		T-shirt	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle Boots
True label	**T-shirt**	**896**	2	15	10	3	1	69	0	4	0
	Trouser	2	991	2	4	0	0	1	0	0	0
	Pullover	15	1	**916**	5	44	0	19	0	0	0
	Dress	9	2	7	**954**	18	0	10	0	0	0
	Coat	0	0	13	19	**928**	0	40	0	0	0
	Sandal	0	0	0	0	0	**989**	0	7	0	4
	Shirt	88	0	43	22	49	0	**796**	0	2	0
	Sneaker	0	0	0	0	0	2	0	**989**	0	9
	Bag	2	1	1	2	2	1	0	0	**991**	0
	Ankle Boots	0	0	0	0	0	6	0	34	0	**960**

TABLE VII. CONFUSION MATRIX OF CLASSIFICATION RESULT WITH FASHION MNIST DATASET USING H-CNN CCP-3 BLOCK MODEL

		Predict label									
		T-shirt	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle Boots
True label	**T-shirt**	**898**	1	17	8	2	1	71	0	2	0
	Trouser	0	**990**	0	6	1	0	1	0	2	0
	Pullover	15	1	**937**	6	19	0	22	0	0	0
	Dress	7	4	8	**952**	12	0	17	0	0	0
	Coat	0	0	23	11	**931**	0	35	0	0	0
	Sandal	0	0	0	0	0	**991**	0	8	0	1
	Shirt	66	0	33	18	40	0	**840**	0	3	0
	Sneaker	0	0	0	0	0	1	0	**990**	0	9
	Bag	4	0	0	3	0	1	0	0	**992**	0
	Ankle Boots	0	0	0	0	0	4	0	26	0	**970**

VII. CONCLUSION

CNN has been applied in a wide variety of fields as a powerful result of the development of deep learning techniques. In fashion application, CNN can support human tasks in image detection, apparel classification, apparel retrieval, and automatic apparel tagging, while the complexity of hierarchy and categories is a challenge in fashion classification. In the past, a hierarchical image classification process was considered in previous studies to improve the accuracy of the classification of apparel. Recently, a hierarchy was used in the Fashion-MNIST data which is 28 ×28 sized grayscale images of 10 classes consisting of 60,000 training images and 10,000 test images, where the Hierarchical Convolutional Neural Network (H–CNN) was proposed in combination with VGGNet architectures (VGG16 and VGG19). Each of these deep VGGNet architectures consists of five building blocks of multiple convolutional, max-pooling, and fully connected layers. However, many filters (in the convolution layer) and many neurons (in the fully connected layer) of each VGGNet (for the Fashion-MNIST data) a required the long training-time.

This study focuses on designing a new efficient architecture for H–CNN to improve not only the accuracy of apparel classification but also the training time. Therefore, the CCP-3-Block architecture was proposed, a shallow-level architecture. A new design combines two convolution layers followed by a pooling layer, designed with only three blocks followed by fully connected layers, and adding batch normalization before the activation function so that networks can learn independently and train quickly, as well as adding the dropout layer in feature extraction and fully connected to reduce overfitting. Moreover, an optimizer (adaptive moment estimation) is added to solve the decaying of gradients, which can improve the overall network performance. In the experiment, the performance was compared of the H-CNN CCP-3-Block model and the original H-CNN VGGNet model (using VGG16 and VGG19 architectures). The results showed that the H–CNN CCP-3Block model performed the better performance with lower loss, higher accuracy, and faster training time than the original H–CNN (VGG16, VGG19) models. This result confirmed the hypothesis that the shallow layered CCP-3 Block architecture performs better performance than the deep VGGNet architectures in the hierarchical classification (H-CNN) of apparel images on the Fashion-MNIST dataset. Thus, for fashion business/applications, the H–CNN CCP-3-Block model can be applied on a variety of real online apparel images with the hierarchical classification.

ACKNOWLEDGMENT

The authors wish to gratefully thank the School of Science, King Mongkut's Institute of Technology Lardkrabang (KMITL), Bangkok, Thailand for the scholarship to Mr. Natthamon Chamnong, a master student in Computer Science, during 2020 – present.

REFERENCES

[1] Y. Taigman, M. Yang, M. A. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2014, pp. 1701-1708.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," nature, vol. 521, no. 7553, pp. 436-444, 2015.

[3] A. Iliukovich-Strakovskaia, A. Dral, and E. Dral, "Using pre-trained models for fine-grained image classification in fashion field," in Proceedings of the First International Workshop on Fashion and KDD, KDD, 2016, pp. 31-40.

[4] A. Iliukovich-Strakovskaia, V. Tsvetkova, E. Dral, and A. Dral, "Non-personalized fashion outfit recommendations," in World Conference on Information Systems and Technologies, 2018: Springer, pp. 41-52.

[5] S. Bhatnagar, D. Ghosal, and M. H. Kolekar, "Classification of fashion article images using convolutional neural networks," in 2017 Fourth International Conference on Image Information Processing (ICIIP), 2017: IEEE, pp. 1-6.

[6] Y. Liu, G. Luo, and F. Dong, "Convolutional Network Model using Hierarchical Prediction and its Application in Clothing Image Classification," in 2019 3rd International Conference on Data Science and Business Analytics (ICDSBA), 2019: IEEE, pp. 157-160.

[7] Y. Seo and K.-s. Shin, "Hierarchical convolutional neural networks for fashion image classification," Expert Systems with Applications, vol. 116, pp. 328-339, 2019/02/01/ 2019, doi: https://doi.org/10.1016/j.eswa.2018.09.022.

[8] F. Chollet, "Keras: The python deep learning library," Astrophysics source code library, p. ascl: 1806.022, 2018.

[9] R. Shanmugamani, Deep Learning for Computer Vision: Expert techniques to train advanced neural networks using TensorFlow and Keras, 1st ed. Packt Publishing, 2018.

[10] N. Raksaard and O. Surinta, "Comparative Study Between Local Descriptors and Deep Learning for Silk Pattern Image Retrieval," Journal of Science and Technology Mahasarakham University, vol. 37, no. 6, pp. 736-746, 2018.

[11] A. Palananda and W. Kimpan, "Classification of Adulterated Particle Images in Coconut Oil Using Deep Learning Approaches," Applied Sciences, vol. 12, no. 2, p. 656, 2022.

[12] K. Xie, L. Huang, W. Zhang, Q. Qin, and L. Lyu, "A CNN-based multi-task framework for weather recognition with multi-scale weather cues," Expert Systems with Applications, p. 116689, 2022.

[13] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, 1998.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," Advances in neural information processing systems, vol. 25, 2012.

[15] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, 2014.

[16] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," arXiv preprint arXiv:1608.03983, 2016.

[17] L. Luo, Y. Xiong, Y. Liu, and X. Sun, "Adaptive gradient methods with dynamic bound of learning rate," arXiv preprint arXiv:1902.09843, 2019.

[18] J. Zhuang et al., "Adabelief optimizer: Adapting stepsizes by the belief in observed gradients," Advances in neural information processing systems, vol. 33, pp. 18795-18806, 2020.

[19] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," Journal of machine learning research, vol. 12, no. 7, 2011.

[20] E. Charniak, Introduction to deep learning. Cambridge, Massachusetts: Random House Publishing Group, 2019.

[21] I. Goodfellow, Y. Bengio, and A. Courville, Deep learning. Cambridge, Massachusetts, 2016.

[22] P. Nuttachot. "Modern Regularization with Augmentation, Batch Normalization and Dropout Techniques." https://blog.pjjop.org/modern-regularization-with-data-augmentation-batch-normalization-and-dropout/ (accessed 19 November, 2021).

[23] Y. Meir, O. Tevet, Y. Tzach, S. Hodassman, R. D. Gross, and I. Kanter, "Efficient shallow learning as an alternative to deep learning," arXiv preprint arXiv:2211.11106, 2022.

[24] Z. Yan et al., "HD-CNN: hierarchical deep convolutional neural networks for large scale visual recognition," in Proceedings of the IEEE international conference on computer vision, 2015, pp. 2740-2748.

[25] X. Zhu and M. Bain, "B-CNN: branch convolutional neural network for hierarchical classification," arXiv preprint arXiv:1709.09890, 2017.

[26] Q. Zhu, T. Hiep Dinh, M. Duong Phung, and Q. Phuc Ha, "Hierarchical Convolutional Neural Network with Feature Preservation and Autotuned Thresholding for Crack Detection," p. arXiv:2104.10511doi: 10.48550/arXiv.2104.10511.

[27] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in International conference on machine learning, 2015: PMLR, pp. 448-456.

[28] M. Hardt, B. Recht, and Y. Singer, "Train faster, generalize better: Stability of stochastic gradient descent," in International conference on machine learning, 2016: PMLR, pp. 1225-1234.

[29] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," arXiv preprint arXiv:1711.05101, 2017.

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.

[31] N. S. Keskar and R. Socher, "Improving generalization performance by switching from adam to sgd," arXiv preprint arXiv:1712.07628, 2017.