SMU Data Science Review

Volume 6
Number 2 *Summer 2022*

Article 4

2022

Hierarchical Neural Networks (HNN): Using TensorFlow to build HNN

Rick Fontenot
Southern Methodist University, rwfontenot@smu.edu

Joseph Lazarus
Southern Methodist University, lazarusj@smu.edu

Puri Rudick
Southern Methodist University, prudick@smu.edu

Anthony Sgambellone
Scientific Test and Analysis Techniques Center of Excellence, Anthony.Scambellone.ctr@AFIT.edu

Hierarchical Neural Networks (HNN): Using TensorFlow to build HNN

Rick Fontenot[1], Joseph Lazarus[1], Puri Rudick[1], Anthony Sgambellone[2]
[1] Master of Science in Data Science, Southern Methodist University,
Dallas, TX 75275 USA
[2] Scientific Test and Analysis Techniques Center of Excellence, 2950 Hobson Way
Wright-Patterson AFB, OH 45431 USA
{rwfontenot, lazarusj, prudick}@smu.edu
Anthony.Scambellone.ctr@AFIT.edu

Abstract. This research demonstrates the use of TensorFlow to build a Hierarchical Neural Network (HNN). Constructing and engineering neural networks to maximize accuracy and efficiency is an active field of research in machine learning. HNN, along with several other applications of split networks have been developed as recently as 2017. However, implementations thus far have required custom-built and coded HNNs. The research conducted here uses TensorFlow to validate this structure by building entirely separate neural nets with logical relations between the output of one net and the inputs of the nets that are downstream. Research has shown that Hierarchical Neural Networks can increase training speed and reduce compute resources. The validation results of HNN using the Fashion-MNIST dataset demonstrate a prediction accuracy of 99.49%, 95.96%, and 88.84% for coarse, medium, and fine level classification respectively, in which the fine level classification accuracy is greater than the baseline model.

1 Introduction

Image classification is a critical tool used for a variety of business and industry domains including medical, automobiles, surveillance and security, online marketing, and social media. Maximizing the accuracy of predictions is often the primary focus, and recent developments in deep learning tools such as Hierarchical Neural Networks provide methods for achieving higher accuracy. For a variety of reasons, deep learning methods are now practical and available to the general public. The core fundamentals of deep learning used today are the result of research from scientists such as Hinton, LeCun, and Li. Based on the work of these researchers it is possible to use deep learning and neural networks in computer vision problems. Images decomposed into pixel vectors can be fed into a neural network and algorithmically extract and identify features needed for classifications. Whereas prior to their work the use of pattern recognition systems in classification problems was not practical for widespread use. The process required specialized skills. Someone had to design a feature extractor to process the raw data to extract the features. This manual process also required that individual to have specific domain knowledge.

In Lecun's 1998 work on document recognition systems and representation learning techniques uses multiple levels to extract features used to classify an image. For example, an image is fed into the model through an array of pixel values. Layer one constructs a crude feature map that distinguishes differences along the edges of the images. The second layer constructs the overarching theme of the image. The key aspect is the feature maps created by the algorithm are not explicitly designed by humans. They are 'learned' or extracted from the data by the algorithm and network procedure.

The widespread adoption of deep learning did not happen immediately following IEEE's publication of LeCun's, "Gradient-Based Learning Applied to Document Recognition" (LeCun et al., 1998). It would take another decade and a half for technological innovation in the areas of data storage, computing, and networking for its use would reach widespread adaption.

Fueling much of the technological innovation over the last 70 years is the ability of the microchip industry to continually fit more transistors on a silicon wafer. Moore's law, named after Gordon E. Moore, who in 1965 observed that the power of microchips doubles about every two years, while the cost is reduced by half. Industry experts debate when this trend will cease to continue. Some suggest Moore's Law will continue through engineering solutions to increase compute power without increasing the number of transistors on the silicon wafer or stacking transistors vertically. Strict interpretations of Moore's Law suggest that the phenomenon is already at its end.

Another key to the success of deep learning was the introduction of graphic processing units. Originally sold to the consumer market for rendering 3D images, GPUs unwittingly found another purpose in scientific computing. Floating-point calculations done in parallel are great for rendering timely graphics in video games. The same hardware in a GPU with different instruction sets will perform multiplication with speeds that greatly outperform CPU thanks to their parallelism as opposed to the core architecture. Moore's law and GPUs democratized better, faster, cheaper computing but the deep learning explosion still needed another key ingredient.

The internet connected the world's computers and allowed people to communicate through their devices. From 1998 through the mid 2010's the use of the internet revolutionized the way people worked, shopped, got their news, and socialized. Greater amounts of data pass through the inter-connected web of computers and live in servers and data centers. Companies soon found that this data was a valuable resource in and of itself. The collection and storage of data on the internet, and its accessibility to a greater amount of people, provided the supply needed for the deep learning revolution.

With the infrastructure in place, the machine learning community saw an explosion in research and application of deep learning techniques. Ecosystems of tools formed such as TensorFlow, Anaconda, and Keras. These resources reduced friction for developers to build and deploy machine learning applications. Thus far, to conduct research into Hierarchical Neural Networks, one has to custom build the models. The ability to call a Hierarchical Neural Networks is not available in the TensorFlow library. This research developed a function to automate model building in TensorFlow. Model Hierarchies are adjusted through a nested dictionary. To make research in this area more accessible this research demonstrates a method for building them using TensorFlow. All code used in this research is publicly available on your GitHub repository.

Neural networks are often deployed in a flat structure. That is each node in the network can be traced along a path to another node. Inputs at the top of the model are fed through layers within the network and finally passed to output layers. This type of architecture assumes all classes are equally difficult to distinguish. Equity in the difficulty of prediction is not always the case. For example, it is easier to distinguish a shoe from a shirt, but it is harder to distinguish a shirt from a jacket. It may be the case that improving the design of the networks produces better results. One such approach to designing networks that address the range in the difficulty of various classifications is Hierarchical Neural Networks.

Hierarchical Neural Networks (HNN) have been studied to distinguish difficult classification problems. The top of the hierarchy is the initial input layer. Here coarse features are extracted by the model. Coarse features are those that are similar across subsequent networks. Subsequent levels of the hierarchy identify more granular differences. Progressively finer classifications are made until the final predictions.

Previous research has demonstrated the value of hierarchical design through improved accuracy compared to flat networks. Improvements in model predictive capability are not the only value in using HNN.

Reduced re-training time in HNN is demonstrated by the research of Liu et al., 2018. The segmentation of the subnets in HNN allows portions of the model to be trained separately (Liu et al., 2018). The segmented structure means only a portion of the subnets needs re-trained at a time. This is particularly useful in the cases when it is desired to add a new category. Another benefit is computational efficiency. Selective execution of the network saves energy and computation time. Mobile applications are constrained by energy and compute power. Devices must be small enough to be portable and have long enough battery life to be practical. Another domain that prioritizes energy efficiency in data centers. Small reductions or increases in energy efficiency compound through the entire system. When attempting to scale in size, data centers are sensitive to the multiplicative effect of energy efficiency. (Liu et al., 2018)

Use cases for hierarchical classification include taxonomy (Bergman et al., 2003), speech-act classification (Kang et al., 2013), music genre classification (Martel et al., 2013), and image classification (Zhou et al., 2022). The March 2020 report from Grand View Research states, 'the market value of image recognition is projected to be over one hundred billion worth by 2027' (Grand View Research, 2020). However, there are only a few approaches to fashion product image classification. The task presents a unique challenge compared to other domains. Fashion products consistently contain multiple attributes and can be arranged in various taxonomies. Fashion is a byproduct of culture and evolves over time and across cultures. This allows fashion items to be parsed into multiple

By 2017 the original 10-class dataset of handwritten digits called MNIST was 20 years old. It is a popular benchmarking dataset for the deep learning community. At this point, the community had moved passed this dataset and needed something more challenging for modern deep learning techniques. Fashion-MNIST dataset was introduced in 2017 with a similar structure but a more difficult alternative (Han et al., 2017). This paper marked the beginning of the Fashion-MNIST dataset becoming the new standard in image classification.

The data set originally came from a German online retailer's website called, Zalando. Images of 10 classes of fashion items; shirts, ankle boots, trousers, coats,

dresses, T-shirts, bags, pull-overs, sandals, and sneakers. Each item appears in a grayscale 28 x 28-pixel image. The 10 classes are balanced each containing 7,000 images for a grand total of 70,000 images. One label is assigned to each image.

Much of the research on the HNN topic, including the hierarchical fashion image classification topic, is done using a model built with customized code. Those interested in building their own HNN require specialized skills. For many buildings, any deep learning model from scratch is impractical, both for reasons of expertise and time. For many data science practitioners, TensorFlow is an excellent tool. Their open-source platforms enable many users to quickly get their deep learning models off the ground.

For all the promising research into HNN, no package or library exists within TensorFlow to implement Hierarchical Neural Networks. This study aims to construct an approach to build an HNN from TensorFlow that can be reproduced by others for their specific data study. If this research proves it can be done, then it would pave the way for further research or the inclusion of an HNN module in TensorFlow. In addition to studying accuracy gains, there will also be a focus on interpreting areas of misclassifications, computing times by structure, and methods to identify sections of the model network for further improvements.

2 Literature Review

The literature review focuses on four areas of neural networks: fundamental research into neural networks, previous research into Hierarchical Neural Networks architecture, convolution vice dense layers, and preventing overfitting.

2.1 Neural Networks

In the beginning, was the perceptron, and the idea of the perceptron was with Frank Rosenblatt. The mathematical model of a biological neuron was thought of in the 1940s, but it was Frank Rosenblatt's papers that inspired the idea of using neuron-like units in computer learning procedures (Rosenblatt, 1962). The perceptron is the main building block on which neural networks can be built. Each node inside a perceptron was a single unit of logic. It is an algorithm that that results in a binary conclusion. Rosenblatt's perceptron model inputs were applied to a biasing weight and then summed. If the resulting sum exceeds a given threshold the perceptron 'fires' which outputs a binary conclusion, one or zero.

Neural networks today differ from the perceptron model introduced by Frank Rosenblatt. Today neural networks can be thought of as an ensemble of linear regression models. Rosenblatt's perceptron model only gave a binary value of zero or one. The introduction of a loss function sometimes referred to as a cost or objective function, meant neurons could give a continuous output. This small difference in structure is what allows modern neural networks 'learn' or minimize the loss function using gradient descent and backpropagation.

2.2 Back Propagation

Fast forward from 1962 to 1986 when Rumelhart proposed the use of backpropagation in neural networks. Backpropagation, or more formally, backpropagation of errors is an algorithm commonly used in deep learning. The algorithm uses gradient descent to minimize the network's loss function. For a single training session, the output of the network is subtracted by the desired output and sum the squared differences. Completing this routine for all batched training examples and adding up the average results gives the total cost of the neural network. Starting from the last layer to the first, the weights and biases of each node are incrementally adjusted to encourage the right nodes to become active when the correct corresponding feature passes through it. Once those changes are discovered the algorithm uses the chain rule to incrementally traverse the network This technique allows the network to iteratively adjust its weights and biases. The gradients at each layer are derived using the chain rule (Rumelhart et al., 1986). The difference between the actual output vector of the net and the desired output vector can be diminished by using the gradient descent to alter each weight by a corresponding amount to the assembled:

$$\partial E \, \partial w \, \partial E \, \partial w$$

Implementation of their work was limited due to the inability to train large neural networks. This was due to slower computers, the way weights were initialized, and limitations of activation functions available at the time.

2.3 Convolutional Neural Networks

The groundbreaking research into Convolutional Neural Networks (CNN or ConvNet) was laid in 1980 in a study of pattern recognition titled Neocognitron (Fukushima et al., 1980). CNN's are the most common type of neural networks used in image classification. They can be used for other tasks, but their success in computer vision problems is what CNNs are known for. CNN's are a type of neural networks with specialized pattern recognition tools called convolution layers. Convolutional layers have filters that power their pattern recognition ability.

Images are represented on a computer screen as a matrix of pixels. The filters in a convolutional layer are a smaller matrix of values that passes over the larger image matrix. The smaller matrix repeats, or strides, over the input space thus reducing the number of parameters to estimate and standardizing how features from the input are analyzed over the input space. The filter matrix is said to convolve across the image matrix hence the name convolution.

Human eyes perceive edges quite well whereas a computer receives a vector of a spatially encoded matrix of pixels. To find the edges within the image, a regularization matrix passes over the larger image matrix. Edges can be thought of as having a direction and magnitude. It is common to first use a gaussian blur first to reduce the low-frequency edges. Then an edge detecting filter such as a sobel operator to detect edges first on a grayscale version of the image. Those features will then be stored in the feature map, which is passed on to the next layer. Each unit in a layer only receives only restricted input from the previous layer. The subsequence layers then combine

these features to extract more complex features or higher-level features, like corners or combinational edges, from the input.

LeCun's research, published in IEEE's 1998 proceedings, demonstrated that a multi-layer network trained with backpropagation could be used commercially to identify handwritten characters (LeCun et al., 1998). They trained a Convolutional Neural Network (CNN) to recognize a complete set of ASCII characters. The data original data set came from the National Institute of Standard and Technology's (NIST) database. From this collection of handwritten numbers, the researchers modified it to suit their machine learning needs creating the well-known MNIST dataset. M for modified and NIST denoting where they got the data. The technology they developed for Bell Labs was used to process checks in the financial industry. What LeCun and his fellow researcher Yoshua Bengio showed in their 1998 paper was that algorithmically derived pattern recognition systems could outperform those constructed by humans using what they called Graph Transformer Networks (GTNs).

2.4 Hierarchical Neural Networks

By the 2010s the internet is ubiquitous, the cost of data storage is much lower, and GPUs are beginning to find their way into scientific computing, which helped make deep learning feasible. This spawned a resurgence in research in the field of deep learning. Amongst the field are a few researchers thinking about the structure of neural networks. They are finding benefits in engineering the structures of the networks.

Neural networks today are commonly composed of three main components the input layer, hidden layer(s), and output layers. In a flat architecture, all the subnetworks are interconnected. The weights for all inputs are distributed throughout one extensive network. In HNN, subnetworks are used to classify in stages.

The Hierarchical Deep Convolutional Neural Networks (HD-CNN) model was proposed in 2014. The HD-CNN follows the classification strategy of coarse-to-fine classes and modular design principles (Yan et al., 2014). It begins with primary CNN coarse classifier utilization to define the coarse class separations, which are easily captured, then moves to the next finer classes. Each subdivided element of HD-CNN deals with distinctive subsets of categories. Throughout the HD-CNN training, the subdivided element pre-training is done, then the global finetuning using the multinomial logistic loss is done. The HD-CNN architecture can be defined by a designer, while each layer element of the CNN may be defined over its training. This study showed that the HD-CNN model performance overcomes the standard CNN.

Three different two-level HD-CNN models with different layer combinations were built to compare the classification performance (Yan et al., 2015). The research evaluated HD-CNN on CIFAR-100 and ImageNet Large Scale 1000-class benchmark datasets and revealed that HD-CNN, in fact, is a flexible deep CNN structure. The team published promising results using their HD-CNN. (Yan et al., 2015).

A research team from Shanghai Jiao Tong University, in collaboration with the University of New South Wales, introduced a variant of CNN called, Branch Convolutional Neural Networks or B-CNN. The B-CNN models establish branches along with the main CNN in order to make multiple predictions ordered from coarse to fine hierarchically. The research team introduces the Branch Training strategy (BT strategy) to train their model. The BT strategy leverages the flexibility of output layers

parameter adjusting to minimize the loss. The data sets used to evaluate their models are MNIST, CIFAR-10, and CIFAR-100. The experiments they ran compared the B-CNN model to CNN Models on the previously mentioned datasets. They demonstrated that the CNN models can be forced to hierarchical classification concepts, and the model performance can be improved with the prior knowledge of hierarchy (Zhu & Bain, 2017).

In 2018, a research team from the University of Michigan introduced Dynamic Deep Neural Networks (D^2NN) to predict images from the Labeled Faces in the Wild dataset and ILSVRC-10 dataset (Liu & Deng, 2018). The D^2NN enhances original neural networks by adding more dynamic decision algorithms resulting in better quality outputs. It can be described as a directed acyclic graph (DAG) without replicated edges. There are three types of nodes in D^2NN: inputs, output, and functional nodes. Its architecture is similar to the conventional DNN, but the control nodes, a special type of function node, can control the computation to skip in some nodes. In this study, the researchers conducted experiments using four different D^2NN structures: a) High-Low, b) Cascade, c) Chain, and d) Hierarchy, as shown in Figure 1. Specifically, this research is interested in the structure and findings associated with structure d) Hierarchy implementation.

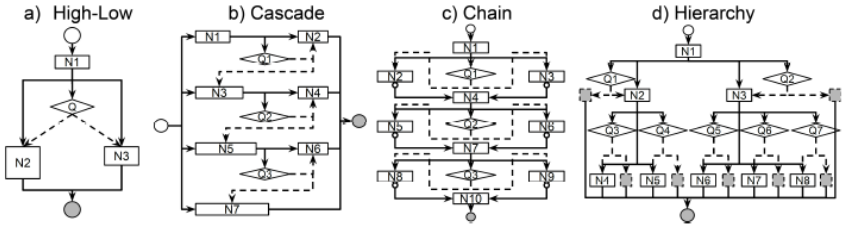


Fig. 1. Four different D^2NN Architectures (Liu & Deng, 2018).

The ILSVRC-10 dataset used in the hierarchy implementation contains 10 different classes; fireboat, gondola, ambulance, jeep, Egyptian cat, Persian cat, Great Dane dog, Deerhound dog, black grouse, and ptarmigan. Each class has 700 images split into 500 images for the train set, 150 images for the test set, and another 50 images for the validation set. One label is assigned to each image. The 10 classes have been categorized into a 3-level hierarchy with 2 coarse-level classes, 5 medium-level classes, and 10 fine-level classes, as shown in Figure 2.

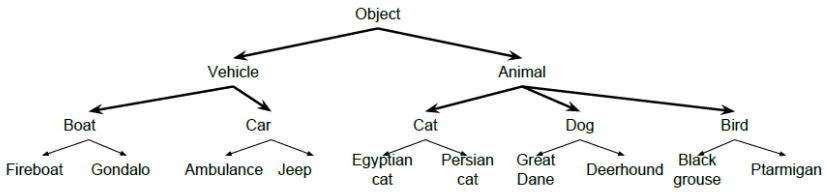


Fig. 2. The ILSVRC-10 semantic class hierarchy (Liu & Deng, 2018).

With the hierarchy implementation in Figure 1 d), an input image starts from the N1 root. Then Q1 determines if it should move downward to N2 and its children or the left branch, while Q2 does similar things for N3 and its children or the right branch. The Q1 and Q2 make decisions independently, which means that an input image can descend parallel to both left and right branches. This multi-threading method prevents the network from prematurely committing to a single path when the input image is ambiguous. The input image then moves down to a branch of Q3 to Q7. At the end of the network, N4 to N8 nodes are in charge of fine-level class classification. The study measured the 4 different D^2NN architectures' performance using F1 accuracy-cost. The results showed that using hierarchical D^2NN reduced half of the computational cost while maintaining the same model accuracy.

In 2019, Researchers Seo and Shin implemented Hierarchical Convolutional Neural Networks (H–CNN) using VGGNet-16 and VGGNet-19, a CNN architecture (Simonyan & Zisserman, 2014), on the Fashion-MNIST dataset (Seo and Shin, 2019). The study aims to solve the difficulties in apparel image classification by using the hierarchical structure on CNN. The way that the researchers organized their H-CNN based Fashion-MNIST is that the top of the hierarchy is the coarse level consisting of goods and clothes. The medium layer assigns labels to some of the classes and passes the rest to a third a final layer. Their H-CNN hierarchy is depicted in Figure 3.

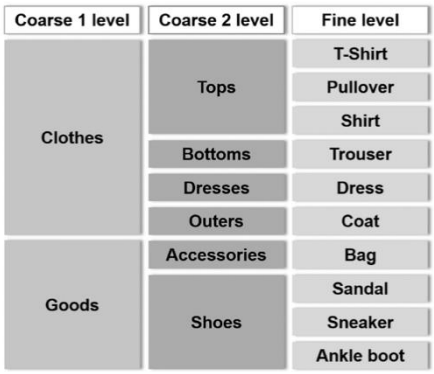


Fig. 3. Hierarchical Classes of Fashion-MNIST Dataset (Seo and Shin, 2019).

The researchers conducted two experiments. First, they compared the VGG16 H–CNN model with the VGG16 base model. In the second experiment, the researchers compared the VGG19 H–CNN model with the VGG19 base model. Their results have shown that the H-CNN models outperform the base CNN models with higher accuracy and lower loss. In conclusion, the hierarchical structure is able to improve the image classification upon the CNN base models.

In 2021, a Queen's University, Ontario Canada research team implemented the use of HNN to classify images from the Kaggle Fashion Dataset by building Condition-CNN on top of the B-CNN. It begins with a standard CNN base model low-level feature

learning (coarse-grained features) across all levels in the class hierarchy. After that, the model learns more specific, higher-level features at each level in an individual CNN subnetwork in parallel. The Condition-CNN accelerates training time and improves the fine-grained class outcomes by providing the predictions priors to the lower level as input features (Kolisnik et al., 2021). They found that a hierarchical structure is efficient because each fine-grained class has only a single coarse-grained class. The use of teacher force eliminates the need to use the Branch Train algorithm. Their research produced the following results. Their CNN achieved the following results on the Kaggle Fashion Dataset; 99.8% accuracy in predicting coarse-level classes, 98.1% accuracy for medium-level classes, and 91.0% accuracy for fine-level classes.

The success of using HNN demonstrates the promise of using hierarchical architecture. Both for their accuracies and efficiency in training and re-training. One obstacle is the lack of a readily available tool in TensorFlow. Google's TensorFlow is an open-source library that allows users to build neural networks easily. It uses a Python API for building applications and executes those in C++. The research is, in part, motivated to demonstrate how TensorFlow can be used to build HNN rather than needing to build or acquire a separate proprietary tool.

2.5 Convolutional Layers versus Dense Layers

In an image classification problem, convolution layers apply a set of filters to an image. Filters are relatively small matrices containing numbers, the size and initialized values within the matrix depend on the type of filter. The smaller filter matrix passes across the larger matrix of the input image. The smaller filter matrix 'convolves' across the image matrix. The dot product of the filter matrix is stored as a convolutional filter of the image. Maxpooling is another kind of filter used in convolutional networks. Same sliding matrix process as before applied to convolutional dot product matrix but maxpooling takes and stores the largest value. In this way, maxpooling reduces the size of the matrix representing the image. When maxpooling is applied, sometimes other techniques of downsampling are used, the resulting matrix stores the activations of the convolutional layer's filters in what is called a feature graph or feature map. Repeatedly applying the same filter to an input result in a map of activations called a feature map that indicates the locations and strength of a detected feature within an image (LeCun et al., 1998). Convolution is effectively a sliding dot product. The kernel shifts along with the input matrix and takes the dot product between the two as if they were vectors.

Fully Connected (FC) layers, sometimes called Dense layers, apply a linear function to the input vector through a weight matrix. All possible connections layer to layer exists, hence the name fully connected or dense layers. In dense layer architecture, every input within the input vector influences every output on the output layer by having some effect on the weight matrix. Dense layers can be used in conjunction with convolutional layers. Picking back up the explanation of convolutional layers, the results of the convolutional filters or feature map is mapped to the dense layer. The sum product of the two matrices is called the dense layer activation.

In an effort to understand the role of dense layers or fully connected layers (FC) researchers Basha et al. performed experiments on 4 different CNN architectures using; CIFAR-10, CIFAR-100, Tiny ImageNet, and CRCHistoPhenotypes datasets (Basha et al., 2020). The result of the research gives guidelines to help understand the tradeoffs

between shallow and deep networks with regards to FC and CNN layers and their performance gains given deeper or wider datasets. The deeper vs. wider data is sometimes referred to as tall-skinny vice short-fat matrices. For a given dataset with an equal number of observations, a dataset is said to be deeper than another dataset if it has more representations per class in the training set (Bansal et al., 2017). Take for example CIFAR-10 & 100. Both Datasets contain 50,000 images. CIFAR-10 is said to be deeper data set because it has 5000 images for each of its 10 classes within the training set. Likewise, CIFAR-100 has 100 classes each containing 500 images and is thus wider.

To summarize the guidelines proposed by Basha et al. Deeper CNNs should be paired with deeper datasets and vice versa. Deep CNN networks will require fewer neurons in their FC/dense layers no matter the type of dataset (Basha et al., 2020).

2.6 Neural Networks overfitting and regularization

Combining the weights from all the parameters can sometimes lead to overfitting. Dropout is a technique used to prevent overfitting. Typically, an overfit model will not generalize well. During the training phase, specifically selected neurons are ignored. They are, in a sense, dropped out randomly during training. This means their contributions to activation functions forward in the network and weights passed backward through the backpropagation algorithm are not applied. Researchers from the University of Toronto published a paper on this topic, "Dropout: An Effortless Way to Prevent Neural Networks from Overfitting" (Srivastava et al., 2014).

Averaging the outputs from many different models is helpful when these models differ from one another. However, training different architectures is difficult and time-consuming to find the optimal hyperparameters for each model. Applying the dropout methodology allows users to approximate many different neural networks architectures efficiently. This significantly reduces overfitting and gives significant improvements over other regularization methods.

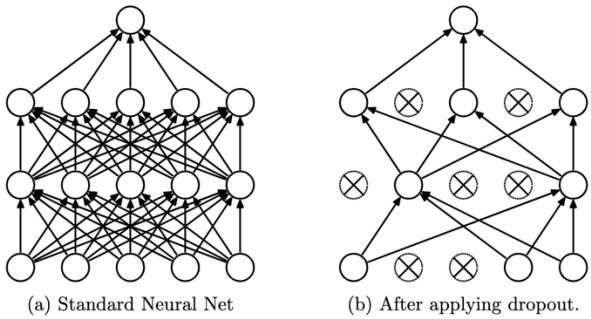


Fig. 4. Dropout Neural Net Model. **Left**: A standard neural net with 2 hidden layers. **Right**: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped (Srivastava et al., 2014).

Another promising regularization technique is VICReg. VICReg, short for variance invariance covariance regularization (VICReg). The variance and covariance terms are responsible for avoiding the collapse problem. While the invariance term is used to make the two output embedding vectors similar. Its use avoids the collapse problem of weight sharing between batches, stop gradient, output quantization, etc. The research shows that variance regularization terms stabilize the training of other methods and lead to performance improvements (Bardes et al., 2021). New advancements in training techniques rely on joint embedding. That creates two identical twins of the networks (Siamese networks) to produce embeddings for different views of the same image. The main challenge of this type of architecture is to prevent the collapse in which two branches ignore the inputs and produce identical and constant vectors. Two methods are typically used to avoid this. Contrastive methods are one technique where one model feeds positive examples and the other negative examples. The problem with the technique is that for higher-dimensional data, the number of contrastive variables is exorbitant. Informative methods such as the quantization-based approaches for the embeddings of different samples to belong to different clusters on the unit sphere. These methods attempt to produce embeddings variables that are de-correlated from each other, thus preventing collapse as described above. The research team applied VICReg on ImageNet using a ResNet-50 backbone pre-trained with VICReg.

VICReg is demonstrated to be useful when two networks are built that provide inference to one another. This structure is called Siamese networks (Pan et al., 2019). The ability of one network to provide inferential information to another network is an important idea for this research into HNNs. After all, HNNs are separate neural networks that feed into branched sub-networks.

While many HNN-based classification and image classification approaches exist, none of these models was built on TensorFlow. The objectives of this research are, first, to implement building HNN-based image classification with TensorFlow without manually constructing the networks to the specific hierarchy, and second, to evaluate if a hierarchical structure of connected sub-neural networks can outperform a flat neural networks model.

3 Methods

3.1 Data

The data used in the study is the Fashion-MNIST dataset, as described in the introduction. The dataset contains 70,000 observations amongst 10 balanced classes. containing Each item contains only one manual label by fashion experts. The 10 classes are balanced each containing 7,000 images for a grand total of 70,000 images split into 60,000 images for the training set and 10,000 images for the test set. Figure 5 shows data samples for all 10 categories with their labels. The dataset has been popularly used as a benchmark for machine learning and deep learning for several years.

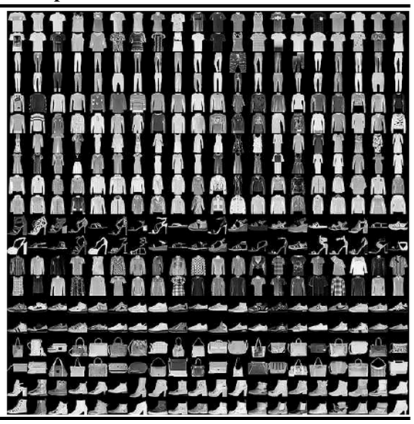
Label	Description	Examples
0	T-Shirt/Top	
1	Trouser	
2	Pullover	
3	Dress	
4	Coat	
5	Sandals	
6	Shirt	
7	Sneaker	
8	Bag	
9	Ankle boots	

Fig. 5. Sample Images with Classification Labels.

Many machine learning libraries have Fashion-MNIST dataset built-in and/or API. The dataset can also be downloaded directly from the Fashion-MNIST GitHub repository (Xiao et al., 2017). For this study, the dataset will be downloaded through TensorFlow.

Table 1. Fashion-MNIST Datasets Sample Size by Classification Categories.

Fine Level		Medium Level		Coarse Level	
Class	Sample Size	Class	Sample Size	Class	Sample Size
T-shirt/Top	6,000				
Pullover	6,000	Tops	18,000		
Shirt	6,000				
Trouser	6,000	Bottoms	6,000	Clothes	36,000
Dress	6,000	Dresses	6,000		
Coat	6,000	Outers	6,000		
Bag	6,000	Accessories	6,000		
Sandal	6,000				
Sneaker	6,000	Shoes	18,000	Goods	24,000
Ankle Boot	6,000				

Multiple hierarchy structures were evaluated. This research uses the benchmark hierarchy structure according to the approach of Seo and Shin, 2019 referenced in section 2.2 above. The count for each fine level, medium level, and coarse level class of the training set is shown in Table 1. The sample counts for each class are intentionally

balanced in this dataset. However, after adding the broader medium and coarse level labels, the class imbalance is an issue of concern in modeling. Each observation in the data contains a 28 x 28 array of pixels that can be plotted for visual representation and flattened for modeling inputs.

3.2 Neural Network Platform and Models

A. The platform used in the research is TensorFlow. TensorFlow is an end-to-end open-source platform for machine learning and artificial intelligence. It was developed by the Google Brain team with the initial version released in 2015 (Abadi et al., 2015). Keras is a high-level application programming interface (API) that is tightly integrated into TensorFlow. It uses TensorFlow as its backend. To answer the research questions and hypothesis, five models had been created with different structures and parameters using TensorFlow Keras. For each model, the training time and classification metrics (accuracy, precision, recall by class) are recorded.

B. Model-1 Base Flat Neural Networks with single dense layer: Built a base model of one neural network with a single dense layer with 128 neurons and 10 epochs to make all classifications as a benchmark for accuracy and training time. This model is hereafter referred to as the "base model". The goal is to create a model definition that can be consistently repeated to compare other model structures rather than to maximize accuracy on this base model. Recorded the training time and classification metrics (accuracy, precision, recall by class).

C. Model-2 Hierarchical Neural Networks with single dense layer: Built using connected individual networks for each branch within the course, intermediate, and fine levels of the Fashion-MNIST relationships. To control for comparison to the base model, each sub-network used the same single dense layer and model conditions as the base model. The classifications from the coarse level model were used to subset the data to feed into separate neural networks for each of the intermediate level groups and similarly those classifications divided the data further into individual fine-layer models. Recorded the training time and the final layer classification metrics (accuracy, precision, recall by class) to compare to the base model. Then compared these results to evaluate whether a hierarchical structure of connected sub-neural networks outperforms the flat neural networks base model in terms of accuracy based on the structure alone. With multiple sub-networks all using a consistent 10 epochs, Model-2 experienced a longer training time than the base model, further models will explore the training time comparisons.

D. Model-3 Flat Neural Networks with early stopping criteria: Expanding on the base model, early stopping criteria was added based on validation accuracy rather than stopping consistently at 10 epochs. Early stopping is based on validation loss with a minimum delta of 0.0001, the patience of 10 epochs, and restores the weights from the best epoch. This model possessed increased training time but better training efficiency. The goal here is to set a benchmark

for training time that can be compared to other models with consistent training criteria.

E. Model-4 Hierarchical Neural Networks with early stopping criteria: Expanding on Model-2, added early stopping criteria based on validation accuracy rather than stopping consistently at 10 epochs. Early stopping matches model 3 and is based on validation loss with a minimum delta of 0.0001, the patience of 10 epochs, and restores the weights from the best epoch. Recorded the training time and classification metrics (accuracy, precision, recall, and F1-score by class).

   a. Multiple hierarchy structures will be evaluated with Model-4A's structure based on the approach referenced in section 2.2 above (Seo & Shin, 2019).

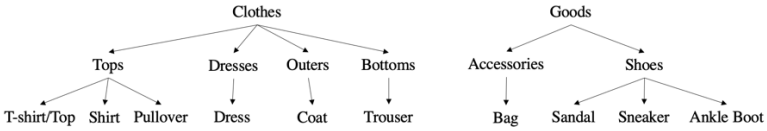


Fig. 6. Hierarchy Structure for Model 4A.

   b. Model-4B has an alternate hierarchy structure developed from insights gained from performance metrics obtained in Models 1 through 4A.

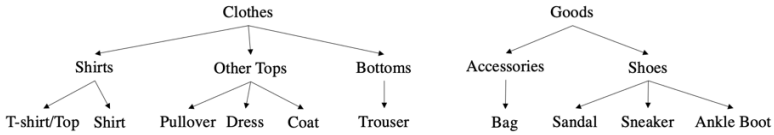


Fig. 7. Hierarchy Structure for Model 4B.

   c. Model-4C has an alternate hierarchy structure developed from insights gained from performance metrics obtained in models 1 through 4B.

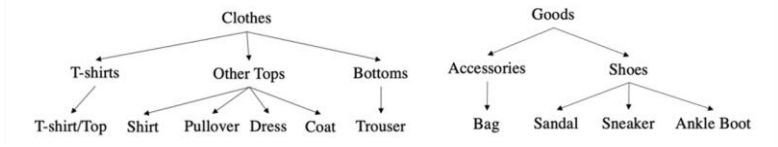


Fig. 8. Hierarchy Structure for Model 4C.

All models' training time and classification metrics (accuracy, precision, recall, and F1-score) were collected. Model-3 and Model-4 give a better comparison of training time for fully trained models versus the comparison base Model-1 and Model-2. Class level precision, recall, and f1 scores were compared for all models for insight into how various structures affect each class. The goal is to determine if a hierarchically structured model can provide classifications suited to the use case, as overall accuracy for all classes may not be the priority when there are costs associated with errors in specific classes.

3.3 Producing Results for comparison

Neural networks use random initial weights and biases which impact the entire training process of a network and the optimization during gradient descent. Therefore, training the same model multiple times can produce different results. One method for comparing different models is to collect predictions over multiple pieces of training of each model and compare the distributions of performance. This method can be computational and time expensive.

The models in this study instead set constant and consistent seeds for all random generators so that each individual model produces repeatable results. All models were trained 3 times to confirm repeatability. Comparisons between different models are done on this basis rather than a distribution of multiple pieces of training. Since models are sensitive to initialization one seed may generate weights that produce better results for one model over another. The effect of different initialized parameters was not studied in this research.

The methods used to achieve repeatability include setting the same fixed seed values in all models for the python environment variable 'PYTHONHASHSEED', the python built-in pseudo-random generator, the NumPy pseudo-random generator, and the TensorFlow pseudo-random generator. The TensorFlow session also has both the intra and inter-parallel threads limited to 1 so that processing methods do not unintentionally introduce differences from training to training.

While these measures allowed for repeatability, it should be noted that the amount of time to train models is affected and could vary from machine to machine. The training time recorded and compared in the results of this study were collected from a single machine thus comparable in relative measures.

4 Results

As referenced above, overall accuracy and training time are critical components in comparing the usefulness of each model. However other tradeoffs such as prediction success for each class may be more important for specific use cases. Examining the performance for each class also helps understand and potentially improve models.

4.1 Overall Accuracy and Training Time

The summary in Table 2 below generally shows that the HNN models performed slightly better than the flat non-hierarchical models in terms of the fine level classification accuracy but at the expense of significantly longer training times.

Paired comparisons demonstrate that the optimized training with early stopping (Models 1 vs. 3 and 2 vs. 4) slightly improved accuracy and doubled training time for both the flat and HNN varieties.

All HNN models performed well separating the coarse level classes with 99.5% accuracy before feeding observations into the medium level models. The best HNN models (Models 4B and 4C) also performed well at separating the medium-level classes before feeding into the final fine-level models. HNN model 4C achieved 96% accuracy at the medium level classifications which is more than 2% better than the flat model at the medium level.

Comparing the fully trained models shows that the initial HNN structure (Model 4A) only gained 0.12% in overall accuracy and tripled in training time vs. the flat NN (Model 3).

Adjusting the hierarchy structure based on discussions in section 4.2 (Models 4B and 4C) improved the HNN's accuracy with only modest increases in training time vs. the original hierarchical structure. Although these HNN's still only increased accuracy by 0.69% and 0.77% respectively over the flat neural network.

Table 2. Accuracy and Training Time By Model.

Model	Description	Training Time	Accuracy by Level		
			Coarse	Medium	Fine
1	Flat NN, 10 Epochs	73s	99.52%	93.74%	87.30%
2	HNN, Structure A, 10 Epochs	208s	99.48%	93.63%	87.88%
3	Flat NN, Optimized Training	133s	99.48%	93.43%	88.07%
4A	HNN, Structure A, Optimized Training	415s	99.49%	93.67%	88.19%
4B	HNN, Structure B, Optimized Training	422s	99.49%	95.16%	88.76%
4C	HNN, Structure C, Optimized Training	444s	99.49%	95.96%	88.84%

4.2 Prediction Performance by Class

All models explored showed varying degrees of prediction success at the fine level classes. Table 3 shows that the flat neural network performed well in most classes but struggled significantly with both precision and recall in the "Shirt" category. The model also has room for improvement in the categories "Coat" and "Pullover."

Table 3. Fine Level Classification Report for Model 3 (Flat NN).

	Model	Class	Precision	Recall	F1-Score
3	Flat NN, Optimized Training	Ankle Boot	0.99	0.97	0.98
3	Flat NN, Optimized Training	Bag	0.83	0.86	0.84
3	Flat NN, Optimized Training	Coat	0.79	0.77	0.78
3	Flat NN, Optimized Training	Dress	0.90	0.88	0.89
3	Flat NN, Optimized Training	Pullover	0.73	0.87	0.79
3	Flat NN, Optimized Training	Sandal	0.98	0.96	0.97
3	Flat NN, Optimized Training	Shirt	0.73	0.62	0.67
3	Flat NN, Optimized Training	Sneaker	0.93	0.97	0.95
3	Flat NN, Optimized Training	T-shirt/Top	0.97	0.95	0.96
3	Flat NN, Optimized Training	Trouser	0.99	0.95	0.97

The first hierarchy structure used (Model 4A) showed improvement vs the flat mode on the "Shirt" category but performed significantly worse on precision and recall for the "T-shirt/Top" category. Similar to the flat NN model this HNN also has room for improvement in the categories "Coat" and "Pullover."

Table 4. Fine Level Classification Report for Model 4A (HNN).

	Model	Class	Precision	Recall	F1-Score
4A	HNN, Structure A, Opt. Training	Ankle Boot	0.95	0.96	0.96
4A	HNN, Structure A, Opt. Training	Bag	0.98	0.96	0.97
4A	HNN, Structure A, Opt. Training	Coat	0.82	0.76	0.79
4A	HNN, Structure A, Opt. Training	Dress	0.87	0.91	0.89
4A	HNN, Structure A, Opt. Training	Pullover	0.76	0.82	0.79
4A	HNN, Structure A, Opt. Training	Sandal	0.98	0.95	0.97
4A	HNN, Structure A, Opt. Training	Shirt	0.72	0.69	0.70
4A	HNN, Structure A, Opt. Training	Sneaker	0.94	0.96	0.95
4A	HNN, Structure A, Opt. Training	T-shirt/Top	0.82	0.85	0.83
4A	HNN, Structure A, Opt. Training	Trouser	0.99	0.97	0.98

Beyond the precision and recall, confusion matrices shown in Figures 9 and 10 were produced to study what misclassifications were being predicted as. The flat NN (Model 3) which struggled the most with "Shirt" shows the misclassifications are spread across predictions of "Coat", "Pullover" and "T-shirt/Top". The HNN (Model 4A) shows that the first hierarchy structure significantly reduces the errors of predicting "Shirt" as "Coat" but has a similar number of erroneous predictions of "Pullover" and "T-

shirt/Top". Based on the hierarchy in Model 4A, the coats were successfully predicted and split off as "Outers" at the medium level and separated from these other categories which fed into the Tops model. This shows that adjustments in the hierarchy can improve an HNNs performance.

Beyond the "Shirt" category, the second most misclassified categories were "Coat" predicted as "Pullover" and vice versa. While the HNN (Model 4A) shows improvements in precision for both categories, it shows a lower recall for both as well. Based on hierarchy structure A, pullovers were classified as "Tops" at the medium level and coats were classified as "Outers", so they were separated before the fine-layer sub-models.

Based on these insights, the hierarchy structure was modified in Model 4B in an attempt to separate the "Pullover" from "Shirt at the medium level as well as include "Coat" and "Pullover" in the same final fine-layer model. Whereas hierarchy structure A included "Tops": {"T-shirt/Top"," Shirt"," Pullover"} and "Outers": {"Coat"}, the new structure in hierarchy B is "Shirts": {"T-shirt/Top"," Shirt"} and "Other Tops": {"Pullover", "Dress", "Coat"}

The original hierarchy structure (Model 4A) also struggled with predicting the "Shirt" category compared to the flat NN. The misclassifications as "Dress", "Pullover" and "Shirt" are the same categories on both flat Model 3 and HNN Model 4A, but the HNN has errors at a higher rate. Since the best performance for "Shirt" was on the flat model, this insight led to another hierarchy structure modification from 4B to 4C where "T-shirt/Top" are separated from all other categories at the medium level and "Shirt" is moved into the "Other Tops" category.

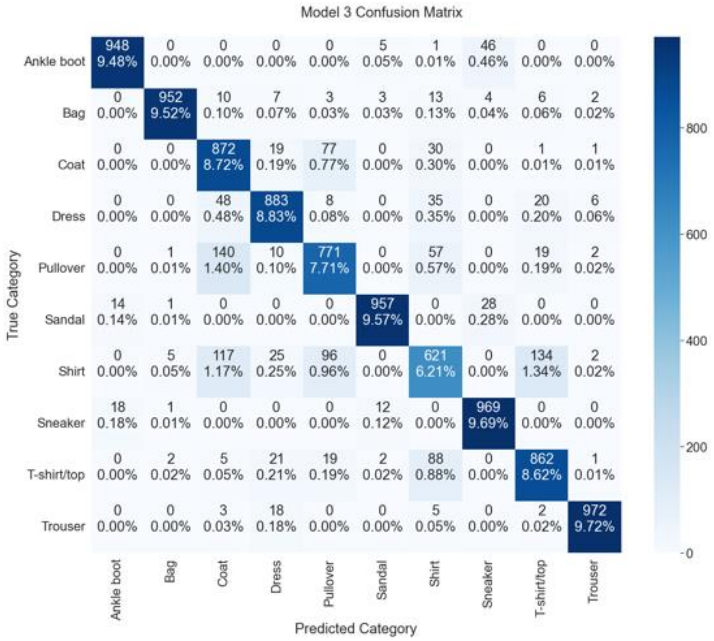


Fig. 9. Fine Level Class Confusion Matrix, Flat NN Model 3.

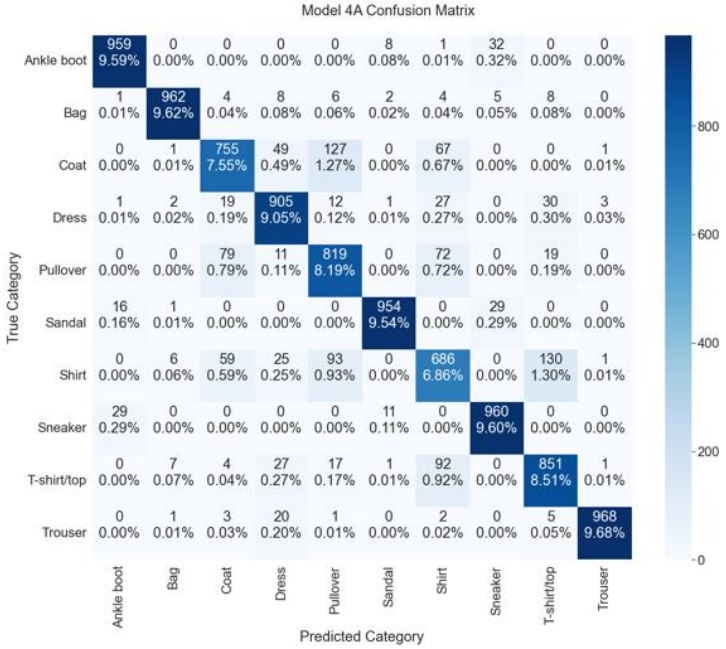


Fig. 10. Fine Level Class Confusion Matrix, HNN Model 4A.

Figures 11-13 compare the precision, recall, and F1-scores for all classification categories across all models. In general, all HNN models significantly outperformed the flat model for the "Bag" category for all three metrics. The flat model significantly outperformed all HNN models for the "T-shirt/Top" category for all three metrics. For all other categories, the HNN models matched or outperformed the flat model and the adjustments to the hierarchy structure through insights discussed above made improvements over the initial structure across all image categories.

The HNN model with hierarchy structure C performed the best of the models explored in this research. This warrants further research in HNN structures to understand their effect on optimization.

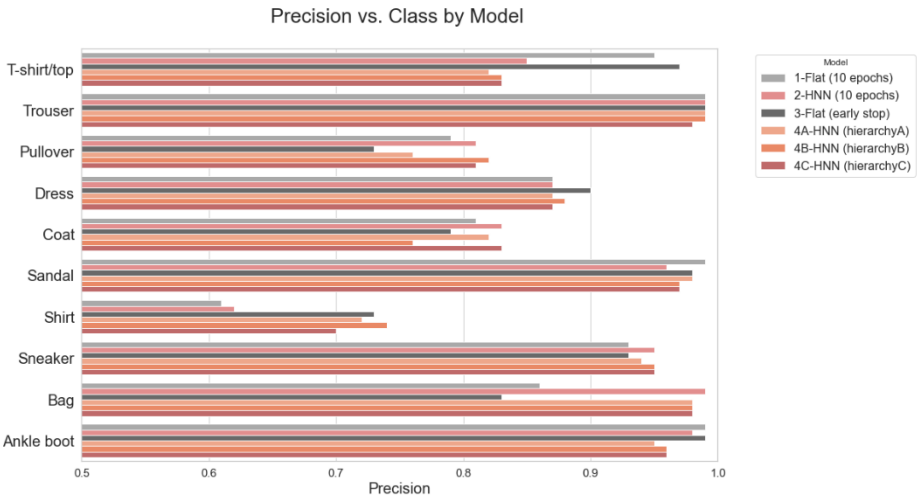


Fig. 11. Comparison of Precision by Class Across All Models.

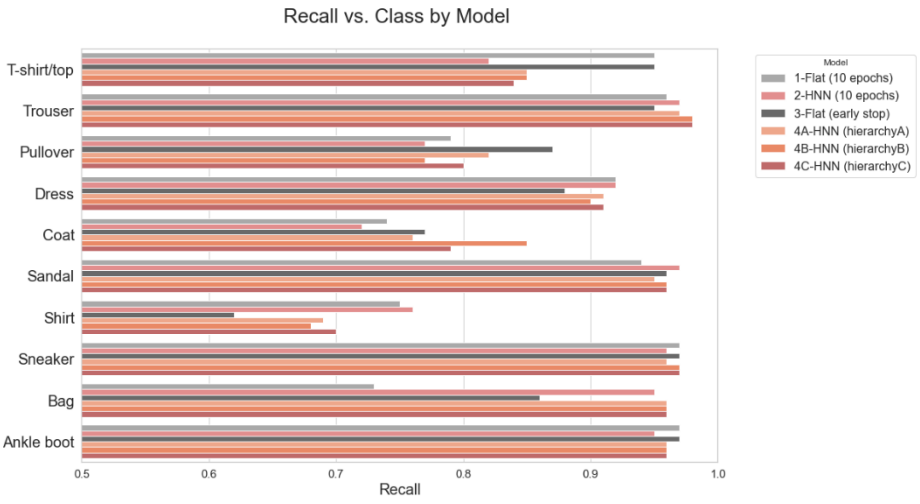


Fig. 12. Comparison of Recall by Class Across All Models.

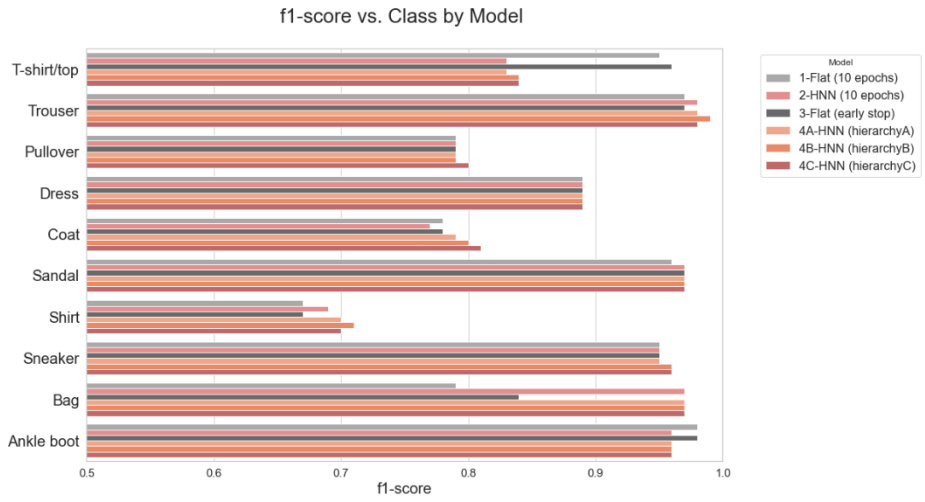


Fig. 13. Comparison of F1-score by Class Across All Models.

5 Discussion

In a direct comparison of HNNs versus flat networks, our research suggests that HNNs offer only small net benefit to overall accuracy though may be able to optimize the performance of a single class.

5.1 Comparison of Results to Other Research

Table 5. Comparison of Related Work Fashion Classification Tasks.

Model	Research Team	Data Set Used	Test Accuracy
VGG16 H-CNN	Seo & Shin	Fashion-MNIST	.935
VGG19 H-CNN	Seo & Shin	Fashion-MNIST	.933
C-CNN	Kolisnik et al.	Fashion-MNIST	.910
C-CNN B	Kolisnik et al.	Fashion-MNIST	.904
TensorFlow HNN-B	Fontenot et al.	Fashion-MNIST	.887
TensorFlow HNN-C	Fontenot et al.	Fashion-MNIST	.888

5.2 Discussion

Comparing the results from this research to related work in fashion classification tasks, the HNN built with TensorFlow does not perform as well compared to Seo & Shin and Kolisnik et al. Direct comparison of results between this research and previous

work ought to be done with the knowledge of how the models were built and with what purpose. The design of the models in this research was done to compare the structure of an HNN to that of a flat NN. Each model was limited to a single dense layer. In addition, this research did not use a convolutional layer. Convolutional layers were used in the other research mentioned in table 5. This is likely the cause of the lower performance and not due to TensorFlow implantation. Given that the models used in this network were intentionally kept simple, the result compared to other researchers are promising.

The results from this study indicate that HNNs do provide advantages compared to flat networks. A direct comparison of the overall accuracy performance of HNNs versus a flat network demonstrates a modest gain in the overall accuracy of 0.77%. An increase in overall accuracy came at the expense of longer training times and greater complexity. When taken into consideration the increase in complexity and training time may not be feasible in many cases. However, for specific use cases, HNNs could be valuable. HNNs segmented structures allow sub-models to be re-trained without altering the results from other models. Another interesting use case is model hierarchy is another parameter users can tune to adjust performance.

Decisions regarding model hierarchy resulted in a dramatic effect on precision and recall scores at the fine level classification. Moving classes in the hierarchy increased the recall and precision of some classes but decreased the results in others. In other words, our findings suggest that model hierarchy is a hyperparameter and could be used in cases were predicting one class takes priority over others.

To illustrate what a priority class is take for example a task in identifying cancer. In this computer vision scenario to identify dermatological markers on the skin for melanoma cancer predictions on this one category take precedence. The model must be excellent at identifying cancer. Correct predictions of other skin lesions, moles, etc. are important but one can understand the singular importance of this one category. Identifying cancer is out of the scope of this research. The example of identifying cancer is used to demonstrate an example of why one might want the ability to prioritize one class in a multi-class problem.

The motivation for moving shirt, coat, and T-shirt classes within the hierarchy were to improve their scores. Model 4A showed signs of difficulty in distinguishing shirts and T-shirts. Thus, in model 4B the research team allowed T-shirts and shirts to pass through their final neural network head-to-head. The rationale being a dedicated network to decipher the minute differences between the two classes ought to improve their lagging scores. This proved not to be the case. Instead, the networks seem to respond in just the opposite manner. Pairing visually dissimilar classes in the same Neural network gives the network greater inferential ability.

Reviewing the results of the HNNs built in this study demonstrates HNNs ability to outperform flat networks in predicting a single priority class. When the classes were moved in the hierarchy, the precision and recall of certain classes changed considerably. Moving shirts and coats, and pullovers to various levels within the hierarchy from 4A, 4B, and 4C resulted in increases in precision and recall in some classes but to the detriment of others. This interesting result suggests that HNNs present a use case for a priority class prediction problem.

5.3 Limitations

The research team's ability to respond to observations on results was limited due to the highly customized and tightly coupled code. To explore HNN using TensorFlow meant the team had to use the library in ways not anticipated by the developers. Thus, deviating from initial test designs to explore other ideas was limited due to time and labor constraints. For example, the Initial experimentation design did not account for the arrangement of classes within the hierarchy as a factor. The phenomenon of the HNNs ability to optimize results on a certain class is drawn from a population of three. In the methods section, setting fixed seeds was discussed as a means of getting repeatable results for each given network. This was useful for a "fair" comparison between these models. Given more time each model could be trained many times to collect distributions of samples on the metrics and tested for statistical significance of performance differences between the models.

5.4 Future Research

The results of the HNN compared to flat networks displayed marginal results, though alternate hierarchical structures were explored. Our research was limited in its ability to make statistical inferences on the effect. Based on our results the research team believes studying the effect of hierarchy on predictions is worthy of future research.

The results of this research did not have as high accuracy when compared to other work in fashion image classification. Reasons for this performance were mentioned in the discussion. To more accurately compare the image classification results from our HNN using TensorFlow to that of other research may require the use of convolutional layers.

5.5 Ethics

Ethics in data science and machine learning is an important consideration. The Fashion-MNIST data set used in this research is a low-threat environment for ethical considerations. One of the reasons Fashion-MNIST is often used for research. In recent years, however, the data science community is becoming increasingly aware of the bias that can unwittingly creep into their work. Algorithms do not possess bias or an agenda per se but the datasets they perform their routines on may be the result of an active biased process.

This research employs a hierarchical structure. Hierarchical representations possess an ability to convey importance or value judgments. When applied to the Fashion-MNIST dataset this framework is innocuous. Decisions on how black and white pictures of clothes flow downward in the hierarchy to receive labels is an antiseptic process. Whether shoes are passed through the 'accessory' network or dresses receive their label from the 'tops' network is trivial. The researchers are not trying to progress a bias against any garment. Yet, if the same hierarchical framework is applied to facial recognition the visual representation of the hierarchy becomes fraught with ethical considerations. Classes such as age, race, gender, and how they appear in relation to others within the hierarchy are the substance for scrutiny. The use of HNNs in future

applications needs to take into consideration labels and their representation within the hierarchy.

Our research uses deep learning and neural networks. These tools can find hidden variables within the data. Even if sensitive data is removed from the dataset before training, factors may be encoded in the data set in other ways. Removing sensitive variables like race or sex from the dataset entirely can prohibit practitioners from identifying biased results. Data science practitioners ought to be aware of the ethical considerations when employing deep learning techniques.

6 Conclusion

The findings from this research show HNNs offer a slight improvement in overall accuracy compared to flat networks. Given the results in this study, the benefits of HNNs do not justify their complexity and additional training time. However, for certain use cases such as a priority class HNNs may offer benefits compared the flat NNs. Moving classes throughout the structure of the model's hierarchy resulted in change at the fine level classification. Based on the finding of this research further investigation of HNNs might demonstrate a use case for optimizing the classification of a priority class.

Acknowledgments. Special thanks to Dr. Jacquie Cheun-Jensen and Nibhrat Lohia for useful discussions and feedback throughout our research.

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., … Zheng, X. (2016). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems.* Retrieved from https://arxiv.org/abs/1603.04467
2. Bardes, A., Ponce, J., & LeCun, Y. (2021). *VICReg: Variance-Invariance-Covariance Regularization for Self-Supervised Learning*. Retrieved from https://arxiv.org/abs/2105.04906
3. Bansal, A., Castillo C., Ranjan R., & Chellappa R. (2017). The Dos and Donts for CNN-Based Face Verification. *Proceedings of the IEEE International Conference on Computer Vision Workshop (ICCVW),* 2545–2554.
4. Basha, S., Dubey, S., Pulabaigari, V., & Mukherjee, S. (2020). Impact of Fully Connected Layers on Performance of Convolutional Neural Networks for Image Classification. *Neurocomputing, 378,* 112-119. https://doi.org/10.1016/j.neucom.2019.10.008
5. Bergman, T., Beehner, J., Cheney, D. & Seyfarth, R. (2003). Hierarchical Classification by Rank and Kinship in Baboons. *Science (American Association for the Advancement of Science), 302(5648)*, 1234–1236. https://doi.org/10.1126/science.1087513

6. Fukushima, K. (1980). Neocognitron: A Self Organizing Neural Network Model for A Mechanism of Pattern Recognition Unaffected by Shift in Position. *Biological Cybernetics, 36(4),* 193–202. https://doi.org/10.1007/BF00344251
7. Grand View Research. (2020). *Image Recognition Market Worth $109.4 Billion By 2027 | CAGR: 18.8%.* https://www.grandviewresearch.com/press-release/global-image-recognition-market
8. Kang, Ko, Y., & Seo, J. (2013). Hierarchical Speech-Act Classification for Discourse Analysis. *Pattern Recognition Letters, 34(10)*, 1119–1124. https://doi.org/10.1016/j.patrec.2013.03.008
9. Kolisnik, B., Hogan, I., & Zulkernine, F. (2021). Condition-CNN: A Hierarchical Multi-Label Fashion Image Classification Model. *Expert Systems with Applications, 182,* 115195-. https://doi.org/10.1016/j.eswa.2021.115195
10. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature (London)*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539
11. LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation, 1(4)*, 541–551. https://doi.org/10.1162/neco.1989.1.4.541
12. Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE, 86*(11), 2278–2324. https://doi.org/10.1109/5.726791
13. Li, P., Li, Y., Jiang, X., & Zhen, X. (2019). Two-Stream Multi-Task Network for Fashion Recognition. *2019 IEEE International Conference on Image Processing (ICIP)*, 3038–3042. https://doi.org/10.1109/ICIP.2019.8803394
14. Liu, L., & Deng, J. (2018). *Dynamic Deep Neural Networks: Optimizing Accuracy-Efficiency Trade-Offs by Selective Execution*. Retrieved from https://arxiv.org/abs/1701.00299
15. Martel, J., Nakashika, T., Garcia, C., & Idrissi, K. (2013). A Combination of Hand-Crafted and Hierarchical High-Level Learnt Feature Extraction for Music Genre Classification. *Artificial Neural Networks and Machine Learning – ICANN 2013*, 397–404. https://doi.org/10.1007/978-3-642-40728-4_50
16. Pan, Z., Bao, X., Zhang, Y., Wang, B., An, Q., & Lei, B. (2019). Siamese Network Based Metric Learning for SAR Target Classification. *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium*, 1342-1345. https://doi.org/1109/IGARSS.2019.8898210
17. Rosenblatt, F. (1962). *Principles of Neurodynamics; Perceptrons and the Theory of Brain Mechanisms.* Spartan Books.
18. Rumelhart, D., Hinton, G., & Williams R. (1986). Learning Representations by Back Propagation Errors. *Nature, 323,* 533-536.
19. Seo, Y., & Shin, K. (2019). Hierarchical Convolutional Neural Networks for fashion image classification. *Expert Systems with Applications, 116,* 328–339.
20. Simonyan, & Zisserman, A. (2014). *Very Deep Convolutional Networks for Large-Scale Image Recognition*. Retrieved from https://arxiv.org/abs/1409.1556
21. Srivastava, N., Hinton, G., Krixhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research, 15,* 1929-1958.
22. Xiao, H., Rasul, K., & Vollgraf, R. (2017). *Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms.* Retrieved from https://arxiv.org/abs/1708.07747
23. Yan, Z., Jagadeesh, V., DeCoste, D., Di, W., & Piramuthu, R. (2014). HD-CNN: Hierarchical Deep Convolutional Neural Networks for Image Classification. *IEEE International Conference on Computer Vision, 2*.

24. Yan, Z., Zhang, H., Piramathu, R., Jagadeesh, V., DeCoste, D., Di, W., & Yu, Y. (2015). HD-CNN: Hierarchical Deep Convolutional Neural Networks for Large Scale Visual Recognition. *2015 IEEE International Conference on Computer Vision (ICCV),* 2740–2748. https://doi.org/10.1109/ICCV.2015.314
25. Zhou, Y., Li, X., Zhou, Y., Wang, Y., Hu, Q., & Wang, W. (2022). Deep Collaborative Multi-Task Network: A Human Decision Process Inspired Model for Hierarchical Image classification. *Pattern Recognition, 124*, 108449–. https://doi.org/10.1016/j.patcog. 2021.108449
26. Zhu, X., & Bain, M. (2017). *B-CNN: Branch Convolutional Neural Networks for Hierarchical Classification.* Retrieved from https://arxiv.org/abs/1709.09890

Appendix

Our source code is available on https://github.com/rickfontenot/HNN-Capstone.