UNSUPERVISED ROUTING STRATEGIES FOR CONDITIONAL DEEP NEURAL NETWORKS

by

Tuna Han Salih Meral

B.S., Computer Engineering, Boğaziçi University, 2017

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering

Boğaziçi University

2022

UNSUPERVISED ROUTING STRATEGIES FOR CONDITIONAL DEEP NEURAL NETWORKS

APPROVED BY:

Prof. Lale Akarun (Thesis Supervisor)	.....
Assist. Prof. Furkan Kıraç	.....
Assist. Prof. İnci Meliha Baytaş	.....

DATE OF APPROVAL: 28.01.2022

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my supervisor, Prof. Lale Akarun, whom I felt blessed to have as my academic guide for her tireless support and guidance throughout my master's journey. It was a great honor for me to work with her. I am grateful to Asst. Prof. Furkan Kıraç and Asst. Prof. İnci Baytaş for participating in my thesis jury and the fruitful discussion.

I would like to thank Ufuk Can Biçici for his endless support. It would be impossible for me to complete this thesis without knowing that there is someone I can ask any question at any time and being sure that I will get an answer.

I would like to thank especially my family for being with me all the time. No word can be enough for me to explain how loved they are.

Getting through my thesis required more than academic support. I would like to thank Sezin Akdeniz for always making me feel special, listening to, and, at times, having to tolerate me. It wouldn't have been as CUTE an experience as this has been without her.

I would like to thank Ferhat Melih Dal for all the brainstorming sessions we have been through, and the dreams we will make happen.

Last but not least, I would like to thank Prof. Aytül Erçil, Ceyhun Burak Akgül, and Erdem Yörük, on behalf of the Vispera family, for the trust and support they bear for me. Vispera is a place that keeps me hopeful for the future.

ABSTRACT

UNSUPERVISED ROUTING STRATEGIES FOR CONDITIONAL DEEP NEURAL NETWORKS

Deep convolutional neural networks are considered state-of-the-art solutions due to their high classification performance in image classification tasks. The apparent drawback is the amount of computing power required to process a single input. To deal with this, this thesis proposes a conditional computation method that learns to process an input using only a subset of the network's computation units. Learning to execute only a part of a deep neural network by routing individual samples has several advantages. Firstly, it is beneficial to lower the computational burden. Furthermore, if images with similar semantic features are routed to the same path, that part of the network learns to discriminate finer differences among this subset of classes, resulting in improved classification accuracy with fewer parameters and computational resources. Investigating the network's activation on a single sample can also help interpret the neural network's prediction. Several works have recently exploited this idea using tree-shaped networks or taking a particular child of a node and skipping parts of a network. In this thesis, we follow a trellis-based approach for generating specific execution paths in a deep neural network. We have also designed a routing mechanism that uses unsupervised differentiable information gain-based cost functions to determine which subset of units in a layer block will be executed for a sample. We call our method Conditional Unsupervised Information Gain Trellis (CUTE). We tested the clustering performance of our unsupervised information gain-based objective function under different scenarios. Finally, we tested the classification performance of our trellis-shaped CUTE network on the Fashion MNIST dataset. We show that our conditional execution mechanism achieves comparable or better model performance than unconditional baselines, using only a fraction of the computational resources.

ÖZET

KOŞULLU DERİN SİNİR AĞLARI İÇİN GÖZETİMSİZ YÖNLENDİRME YÖNTEMLERİ

Derin evrişimli sinir ağları, imge sınıflandırma probleminde yüksek performansları nedeniyle en gelişmiş çözümler olarak kabul ediliyor. En belirgin dezavantajları, tek bir girdiyi işlemek için gereken yüksek işlem gücü ihtiyaçları. Gerekli iş gücünü azaltabilmek için, bir girdiyi sinir ağının hesaplama birimlerinin yalnızca belli bir alt kümesini kullanarak işlemeyi öğrenen koşullu hesaplama yöntemleri bulunmakta. Girdileri yönlendirerek derin bir sinir ağının yalnızca bir bölümünü kullanmayı öğrenmenin birçok avantajı var. İlk olarak, işlem yükünü azaltmanın büyük bir fayda sağlayacağı aşikardır. Ayrıca, benzer özelliklere sahip imgeler aynı yola yönlendirilirse, ağın belli bir kısmı bu sınıflar arasındaki daha ince farklılıkları ayırt etmeyi öğrenir ve bu da daha az parametre ve işlem gücü ile daha iyi sınıflandırma başarımı sağlar. Yapay sinir ağının belli bir girdi karşısındaki aktivasyonlarını inceleyebilmek, sinir ağının tahminlerini yorumlamaya yardımcı olabilir. Son zamanlarda bazı çalışmalar, ağaç şeklindeki ağları kullanan veya bir düğümün belirli bir çocuğunu seçerek ağın bazı parçalarını es geçmeyi öğrenen koşullu öğrenim modelleri önerdi. Bu tezde, derin bir sinir ağında belirli yolları kullanmayı öğrenmek için kafes yapısı temelli yeni bir yaklaşım izledik. Ayrıca, bir imge için bir katman bloğundaki birimlerin hangi alt kümesinin kullanılacağını öğrenen denetimsiz türevlenebilir bilgi kazanımı tabanlı bir yitim fonksiyonu kullanan yeni bir mekanizma tasarladık. Yöntemimize Koşullu Denetimsiz Bilgi Kazanımı Kafesi (CUTE) diyoruz. Denetimsiz bilgi kazanımı tabanlı yitim fonksiyonumuzun kümeleme başarımını farklı senaryolarda teste tabi tuttuk. Son olarak, Fashion MNIST veri kümesi üzerinde CUTE mimarimizi denedik. İşlem gücünün yalnızca bir kısmını kullanan koşullu öğrenim mekanizmamızın, koşullu öğrenme kullanmayan referans modellere karşı karşılaştırılabilir veya onlardan daha iyi başarım sağladığını gösterdik.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

$\|A\|$	Cardinality of set A
$\{x, y\}$	Data point
`argmax`	Argmax function
$\mathbb{E}$	Expected value
F_i	i^{th} layer
$F_{l,i}$	i^{th} computational block at the l^{th} layer
$F_L \circ F_{L+1}$	Compositional sequence of computational nodes
H	Router Network
$\mathbb{H}[p(x)]$	Entropy of the probability distribution
IG	Information gain
J_C	Objective function
L_{CUTE}	Global loss of CUTE network
$\mathbb{MI}[U, V]$	Mutual information between set U and set V
$\mathbb{NMI}[U, V]$	Normalized mutual information between set U and set V
$p(x\|y)$	Conditional probability of x given y
$\mathbb{RI}$	Rand Index
U	Ground truth labels
V	Predicted labels
Z	The matrix indicating the expert selection
θ	Neural network parameters
$\frac{\partial y}{\partial x}$	Partial derivative
λ	Weight parameter
ϕ	Routing network parameters

LIST OF ACRONYMS/ABBREVIATIONS

CIGN	Conditional Information Gain Network
CNN	Convolutional Neural Network
CUTE	Conditional Unsupervised Information Gain Trellis
DAG	Directed Acyclic Graph
NMI	Normalized Mutual Information
PCA	Principal Component Analysis
ReLU	Rectified Linear Unit
t-SNE	t-distributed Stochastic Neighbor Embedding
UIGC	Unsupervised Information Gain Clustering

1. INTRODUCTION

Image classification is one of the most fundamental problems in the computer vision domain, and deep learning models have become the state-of-the-art approaches in image classification since the introduction of AlexNet [1] in the ImageNet [2] 2012 challenge with an error rate nearly 10.8% lower than the runner-up model. Since then, deep learning models have become more accurate while getting larger and heavier computationally, size-wise, and time-wise. This makes these models impractical for edge computation and infeasible for embedded devices.

Conditional computing has offered to tackle the problems introduced by computationally expensive models by allowing the activation of a subset of the network dependent on the input, which directly reduces the computational burden of the architecture. One of the most notable advantages of computational methods is that they can allocate computations on-demand at test time by selecting only a subset of the whole model. Another direct advantage is that subsets of the network activated by a particular type of inputs become better at classifying those types since the gradients will be sharper and more distinctive. In addition, conditional methods allow a trade-off between accuracy and efficiency. Therefore, they are more adaptable to low-computation environments and embedded devices. There are similar mechanisms in the human brain, which are believed to process information in a dynamic way [3]. It is possible to investigate which part of the network is activated given an input sample and which parts of the input are accountable for specific predictions.

Conditional networks that use routing to handle dynamic activation of a sub-network have gating mechanisms that selectively employ branches based on the input sample and use the selected branch's transformation to the input sample for the next layer. These types of models can have tree-shaped or trellis-shaped architectures, as seen in Figure 1.1. A similar idea has been exploited in "Conditional Information Gain Networks" (CIGN) [4], which builds a tree-structured network and presents router net-

works for each non-leaf node. In CIGN, the router mechanisms trained with supervised local information gain by learning an optimal partition, such that the data routed into the subtrees become more purified, in the sense that semantically similar classes are brought together. Trainable weight parameters in the corresponding subtrees learn discriminative representations for their respective data groups, downsizing the number of parameters in the tree branches. The problem with the tree structure is that misrouted samples can end up in subtrees that are not specialized to classify them. When a sample is misrouted in this manner to the wrong branch, it is possible to be misclassified. The trellis structure has the advantage of misrouted samples recovering from misrouted paths in later stages.

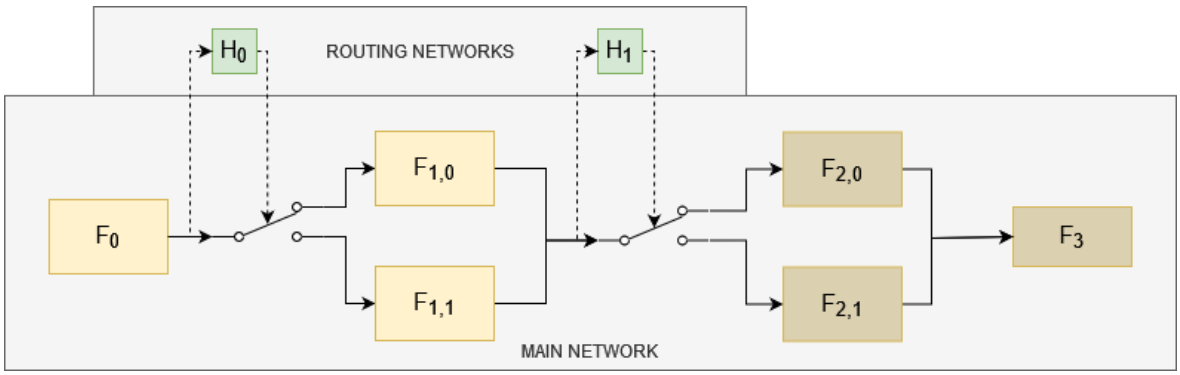

Figure 1.1. A trellis-shaped architecture.

1.1. Contributions

In this thesis, we propose a trellis-based approach for generating specific execution paths in a deep neural network. Our novel trellis-shaped model allows misrouted samples to recover. In a tree, there is only a single path between two nodes. However, there are multiple paths between two nodes in a Trellis structure. If a sample is misrouted to the incorrect computational unit, it will still have a chance to select the correct path in the next layer. In this work, we propose Conditional Unsupervised Information Gain Trellis, CUTE, which uses an unsupervised version of information gain-based routing mechanism in [4] in a DAG or Trellis structure. Information gain-based losses in each layer of the Trellis structure lie at the model's heart. We aim to

create expert paths in the Trellis structure for different semantic groups in the given data. Given a convolutional or fully connected layer or a single block of such layers, we divide the computational units into N parallel groups (called F units), whose aim is to learn required representations for the data and a single H unit in layer l, H_l, called a router, which learns a probability distribution over the F units of layer $l+1$, F_{l+1}. The probability distribution on each block's F units defines a global probability distribution on each root to leaf path in the Trellis structure, and hence the model can be treated as a hierarchical mixture of experts model [5]. The mixture of experts model does not lead to a straightforward EM training due to the large number of experts and each root-to-leaf expert being a convolutional neural network. Instead, we define routing mechanisms based on differentiable information gain objectives [6], which allow sparse execution of the computational blocks and group samples having semantically similar features together. When similar samples with minor semantic differences follow the same root-to-leaf paths in the DAG structure, it can be hypothesized that the deep network making up the root-to-leaf path will learn to differentiate between the finer details of such samples, automatically converting the original multi-class problem to a finer one.

An example diagram for a CUTE model is in Figure 1.1. In this figure, F nodes are used for feature extraction, while H nodes use as router networks that decide which one of the parallel F nodes will use. F nodes may contain a single dense layer, a convolutional layer, or the combination of several dense, convolutional, and activation layers. H nodes may also be composed of several layers, but we try to keep the node sizes minimum to avoid the additional computational burden.

The probability distributions over each block's F units are trained with the previous block's corresponding unsupervised local information gain losses. We hypothesize that the Trellis structure is inherently a more robust topology than trees, presented in [4] since when a sample is misrouted in a particular block, it can be recovered in the blocks following it.

1.2. Thesis Outline

The rest of the thesis is structured as follows: In Chapter 2, we present a literature survey of the existing methods and models to summarize current and previous studies in the context of our research on conditional and dynamic computation in deep neural network architecture. In Chapter 3, we give the detailed formulation of CUTE and explain CUTE training and inference methodology. In Chapter 4, we present the experiments. In the first section of this chapter, we try to illustrate the effectiveness of the unsupervised information gain objective on the clustering task. Then, we investigated our method's effectiveness and routing behaviors on the well-known Fashion MNIST [7] dataset. Chapter 5 concludes the thesis.

2. CONDITIONAL COMPUTATION METHODS

Image classification with deep learning models firstly focused on the well-known MNIST [8] dataset for the digit classification problem, unseating the SVM-based methods in 2006 [9]. In 2012, deep learning models became the state-of-the-art approaches for the image classification tasks on ImageNet [2] by a large margin compared to the runner-up methods [1]. The challenging part was scaling from MNIST, a dataset containing 28×28 grayscale images, to ImageNet, consisting of 256×256 RGB images. The deep models that handled the challenge employed the convolutional layers whose sequential nodes were not fully connected but only sparsely connected. They have also utilized the GPU technology to speed up training procedures by parallel computation [10]. As we inspect the ImageNet classification performances of the neural networks, we can see larger models performing better provided the appropriate regularization methods are applied. Therefore, we can say that it becomes a new challenge to create efficient models as well as well-performing large models. As mentioned in [11], conditional computation methods can be a path to creating efficient network models with increased performances.

Convolutional Unsupervised Information Gain Trellis (CUTE) model can be categorized and compared with various areas in deep learning literature. Firstly, it is most certainly a conditional computation model since the network selectively activates a part of the network depending on the input sample. CUTE also consists of sequential and parallel compositions of several Mixture of Expert nodes, which lets our model be categorized as a Hierarchical Mixture of Experts Model. Since the CUTE architecture has a trellis-shaped graph structure, it is possible to compare CUTE with decision tree-neural network hybrid models.

2.1. Conditional Neural Networks

Conditional computation in neural networks aims to reduce the computational cost during inference and training by selectively activating a subset of the network. It can additionally try to allocate a subset of the network for a subset of samples corresponding to a specific subset of the input space. One of the pioneering works in conditional computation for neural networks is presented in [12], which investigates several approaches to estimate the gradient of a loss function with respect to stochastic neurons that can be switched on or off according to their outputs.

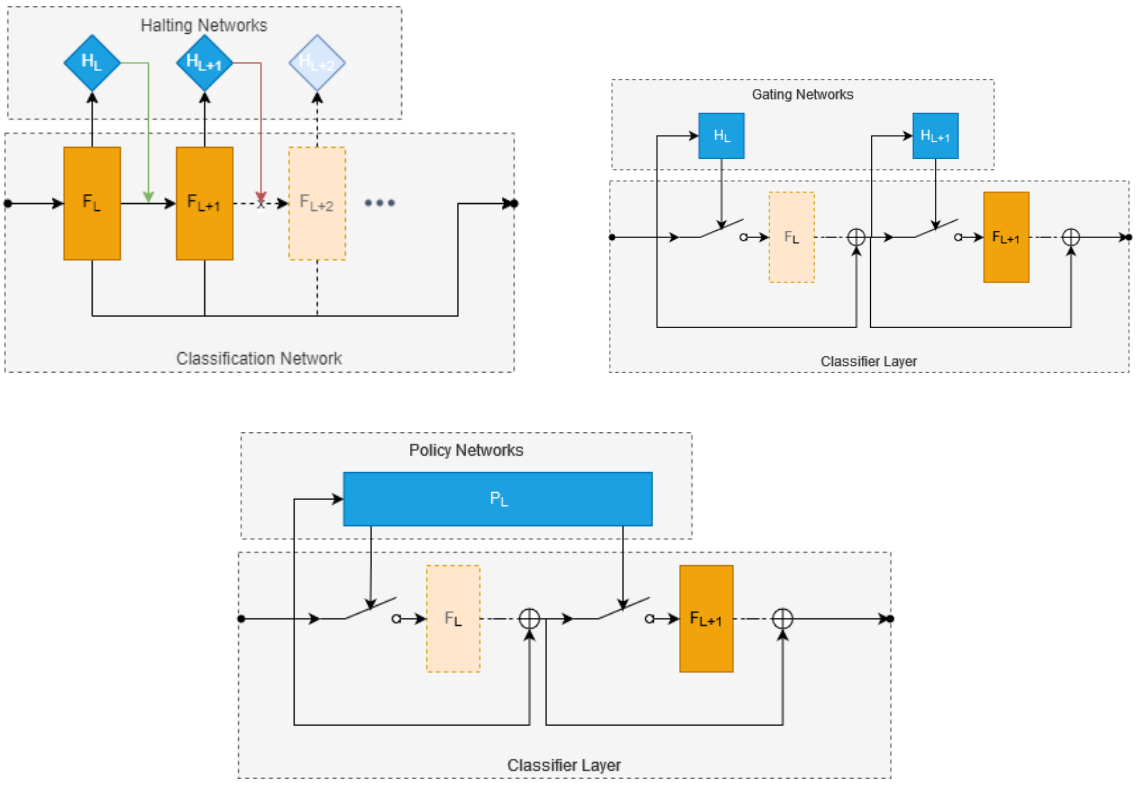

Figure 2.1. Example dynamic layer skipping architectures. Upper Left: After F_{L+1}, the halting network decided to stop execution. As a result, layers F_{L+2} and beyond are not executed. Upper Right: Figure shows the gating module, which determines the execution of a block based on the output of the preceding layer. Lower: The skipping choices for all layers in the main network are generated by the policy network.

Murdock et al. [13] hypothesized that parallel layers could be represented as a single, combined layer with a block-structured weight matrix. Their work is a generalization of Dropout [14], which allows for structure learning via back-propagation to parameterize hierarchical architectures. Bengio et al. [15] proposed a conditional computation method utilizing reinforcement learning approaches. They attempt to preserve the prediction accuracy of the network while sparsifying the network activations.

BlockDrop [16] uses residual connections that activate according to the sample. Instead of executing all the blocks of a ResNet architecture, BlockDrop learns an activation policy using reinforcement learning to select the minimal configuration of blocks needed to classify an input sample. They argue that the resulting sample-specific paths in the network reflect the input's difficulty and encode meaningful visual information. That is, easier samples activate fewer paths, while some paths are activated only to extract visual patterns used to distinguish a cluster of classes. A similar approach to conditional computing is presented by Veit et al. [17] as ConvNet-AIG. The architecture of ConvNet-AIG is similar to BlockDrop, but gates decide the activation of each layer. The gates that decide to execute or skip a layer are trained end-to-end with the network as opposed to BlockDrop, which requires additional training procedures that include reinforcement learning. In a similar work, Wang et al. proposed SkipNet [18], a modified version of ResNet that uses gating networks to decide to skip blocks of the network based on the activations of the previous layer. They use a hybrid training procedure combining supervised learning and reinforcement learning to handle the nondifferentiable hard-gating problem.

McGill et al. [19] proposed a cascaded network architecture that consists of a multi-column multi-row architecture. Their approach is to stop network activations at an early layer and pass ambiguous samples forward to subsequent layers for further execution. They offer different training procedures for their architecture. Figurnov et al. [20] proposed a ResNet-based architecture that dynamically configures the number of layers executed for each input sample. They hypothesized that the network, SACT,

stops the computation at the layer where the extracted features are good enough for classification. Liu et al. [21] proposed dynamically throttleable neural networks, TNN, which adaptively manage the balance between the target performance and resource usage. Cai et al. [22] proposed Dynamic Routing Networks, DRNet, which execute layers depending on the instances using lightweight routing networks generating branch importance weights for each branch. They also use the Gumbel-Softmax [23, 24] to approximate sampling discrete branch selection.

Ioannou et al. [25] hypothesized that applying a block-diagonal pattern of sparsity to the activation between layers in a neural network is equivalent to giving the network architecture a routed tree-shaped structure. They created a conditional architecture beforehand during training and used the learned structure in inference a priori. Teerapittayanon et al. [26] introduced BranchyNet, which allows early exiting mechanisms for known architectures such as AlexNet and ResNet. They hypothesized that features extracted from the early layers are enough for good classification performance for most of the input, while complex samples can be routed forward to extract more features. They added that exiting early enables the network to create better gradients for the early layers. Inspired by the left-right asymmetry of the brain, Jiang et al. [27] proposed a two-branch solution, one of which predicts the coarse labels while the other branch makes fine predictions. The two branches include mechanisms for skipping layers using a minimal gating network.

2.2. Mixture of Experts Methods

A machine learning algorithm that splits a problem space into homogenous parts using multiple learners is called a mixture of experts. Jordan et al. [5] proposed an architecture called Hierarchical Mixtures of Experts where multiple levels of gating functions are used to condition model output. Gating mechanisms assign weights to each expert in their model, and their predictions are accumulated. The learning algorithm of gating functions is based on the Expectation-Maximization algorithm. The mixture of experts approach is not preferable in the deep neural network domain

due to the high computational burden for training and evaluation. However, Eigen et al. [28] proposed Deep Mixture of Experts, DMoE, employing gating networks at each layer of a multilayer network. They hypothesized that by conditioning the gating and expert networks on the output of preceding layers, their model can provide an ensemble combining an exponentially large number of valuable experts.

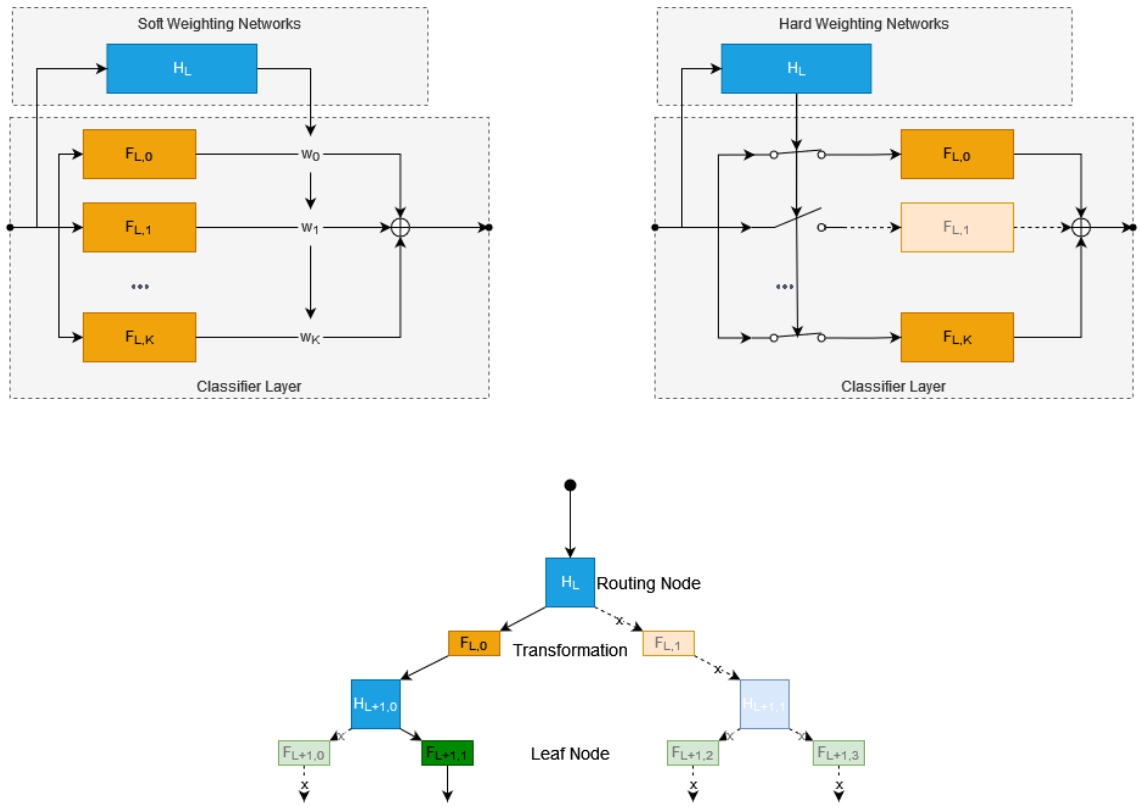

Figure 2.2. Example Mixture of Experts and Tree architectures. Upper Left: Network with soft weighting schemes employ auxiliary modules to generate the weights for branches. Upper Right: Network with hard gating mechanisms uses additional networks to make hard gating decisions. Lower: In the tree structure, routing and transformation nodes are represented respectively.

Shazeer et al. [29] introduced a neural network component consisting of several sub-networks controlled by gating networks that determine which sparse combination of experts will be used. Ahmed et al. [30] proposed a network comprising of a two-stage training scheme. Firstly, the network is trained to classify coarse labels learned through

training. Secondly, the early network layers learned in the first stage are used as initial weights. An expert for each coarse label is trained to discriminate the fine classes belonging to the coarse label. Yan et al. [31] proposed a similar method. They used coarse labels determined beforehand for an easy classification task, and challenging classes were routed to fine classification experts.

2.3. Neural Network-Decision Tree Hybrids

Our CUTE architecture uses a trellis-shaped architecture, but decision trees from different perspectives inspire it. The routing strategy of CUTE depends on an information gain-based loss function similar to the splitting criteria of some decision trees. Murthy et al. [32] proposed Deep Decision Networks where the network is built in multiple stages during training. The network creates a classifier for the easily distinguished classes at each stage and partitions the remaining data. All the weights in the previous layers are frozen while training in the next stage. The resulting network has a binary tree-shaped architecture. Kontschieder et al. [33] proposed a network that unifies the decision trees and deep convolutional networks. They utilize the representation learning power of the convolutional layers at the early stages of the network to provide valuable features for the splitting functions and finalize the network with differentiable decision tree classifiers. Zhou et al. [34] introduced Multi-Grained Cascade Forest, gcForest, which consists of several layers of random forest classifiers instead of differentiable neural network layers.

Biçici et al. [4] proposed Conditional Information Gain Networks, CIGN, that built a tree-structured architecture and introduced router networks at each non-leaf node trained using an information-gain-based objective function. They intend to create expert nodes in the tree to classify a subset of similar classes. At each level of the tree, the distribution of classes becomes purer, and more refined classifiers are trained at the leaves of the network. Overall, the network is trained end-to-end using a combination of routing and classification objectives. Our work is an improvement over the architecture and the routing objective of the CIGN.

3. CONDITIONAL UNSUPERVISED INFORMATION GAIN TRELLIS

3.1. Architecture

The CUTE algorithm builds upon a Trellis structure of computational units. A trellis structure (Figure 3.1) is a graph split into vertical slices in which each node in each slice connects to at least one node in the earlier slice and at least one node in the next slice. The nodes in the earliest slice have connections to only the next slice, and the nodes in the latest slice in the trellis have connections only to the earlier slice.

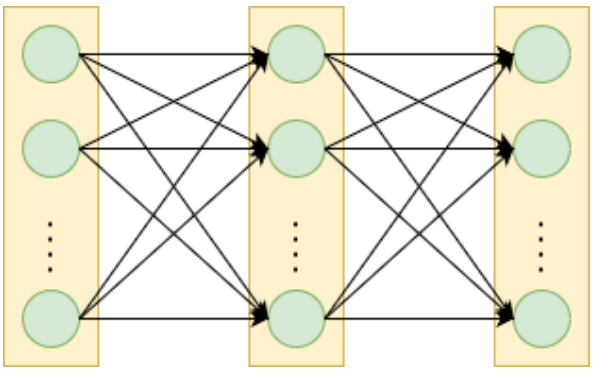

Figure 3.1. Trellis Graph Structure.

In this thesis, we propose a conditional architecture that uses an information gain objective to route selected subsets of the network. Since we use unsupervised techniques to compute the information gain objective, we call this the Conditional Information Gain Trellis (CUTE).

In the CUTE, a feed-forward deep neural network is divided into blocks consisting of sequential layers F_i. To extract low-level features shared by all subsequent layers, the CUTE architecture has a certain amount of constant computation in block 0, F_0. Each F_i, $0 < i < N - 1$, has K_i parallel processing units. We divide layers of every block in the feed-forward network into K_l parts. An example architecture based on the

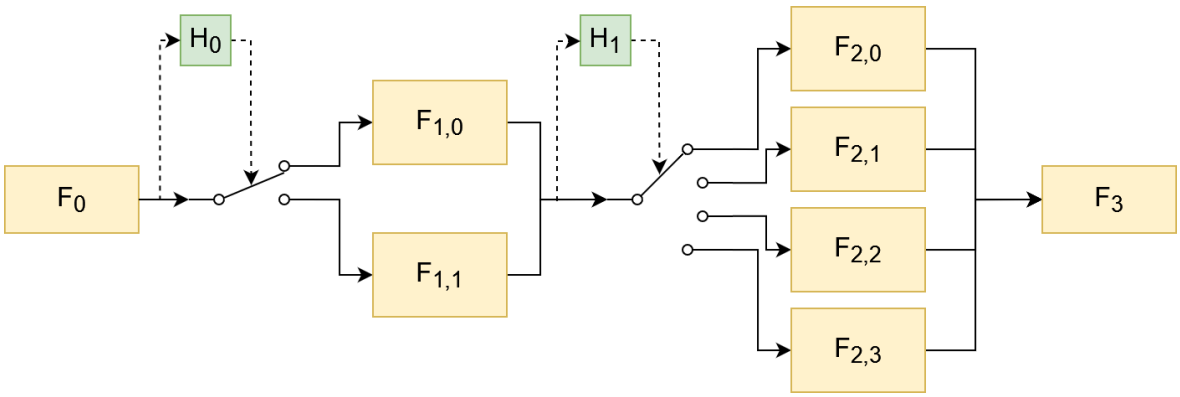

Figure 3.2. An example CUTE architecture.

LeNET architecture [35] is presented in Figure 3.2 If the block has a convolutional layer with C filters, we use C/K filters in each parallel unit. Similarly, if a block has a fully connected layer with D nodes, we use D/K filters in each parallel unit. Although we could have used different block structures in parallel units inside each block, we opted to use identical units for each parallel unit. Example conversion from a conventional feed-forward network block to a CUTE block is presented in Figure 3.3

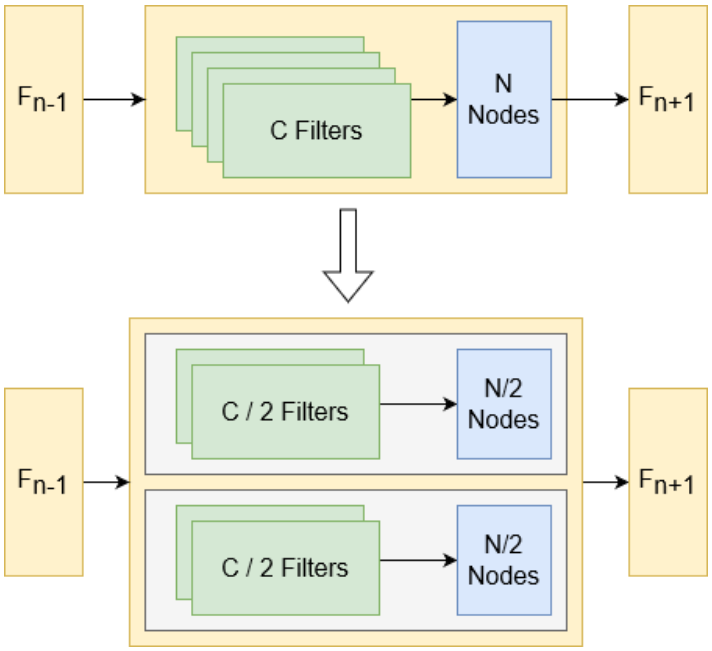

Figure 3.3. An example conversion from a conventional feed-forward Network block to CUTE block.

Each parallel unit in a CUTE block can contain variable number of convolutional layers, fully connected layers, which can also have deep neural network operations such as Batch Normalization [36], Layer Normalization [37], Dropout [14], and different types of non-linear activation. We call the i^{th} computational block at the l^{th} layer as $F_{l,i}$. Every feed-forward operation in the CUTE utilizes only one block in each layer. Therefore, a compositional sequence of computational nodes $(F_L \circ F_{L-1,i_{l-1}} \circ \cdots \circ F_{2,i_2}, F_{1,i_1}, F_0)$ makes for a root-to-leaf expert. Each such root-to-leaf expert defines a posterior distribution $p(y|Z, x, \theta)$ for the given data point $\{x_i, y_i\}$. Z is defined as the matrix indicating the expert selection for each layer. Row l of Z shows the expert selection for layer L_l. Each row in Z is a one-hot vector. For example, for $K_l = 3$ at layer l, if we have $Z_l = [0, 1, 0]$, this means that the computational block $F_{l,1}$ is activated while $F_{l,0}$ and $F_{l,2}$ are skipped for the given sample.

Each computational layer in the CUTE includes an additional routing unit H. H units decide which parallel blocks each sample will follow in the next CUTE layer. They are connected to outputs of the F blocks in the same layer and parameterize a probability distribution $p(Z_{l+1}|x, \theta, \phi)$ on the next layer of F blocks. θ contains all neural network parameters such as convolutional weights and fully connected layer weights driving the F blocks. ϕ contains all parameters which transform the output of F blocks into a probability distribution in the H units. Since H units take the outputs of F blocks as input, it depends on the θ parameters. F blocks are trained with the categorical cross-entropy loss, while H parameters are trained with the unsupervised information gain loss.

Given a dataset $\{x_i, y_i\}_{i=0}^N$, the classification objective of the mixture of experts model can be described as

$$\mathbb{E}_{p(x,y)}[\log p(y|x, \theta, \phi)] = \frac{1}{N} \sum_{i=1}^{N} \log \sum_{Z} p(y_i|Z, x_i, \theta) \prod_{l=1}^{L} p(Z_l|x_i, \theta, \phi). \tag{3.1}$$

It is challenging to calculate stochastic gradient estimates for the objective func-

tion in Equation (3.1). Therefore, we will use the lower bound of the objective, similar to the approach of REINFORCE, [38] which makes use of Jensen's inequality as

$$\begin{aligned} J_C(\theta, \phi) =& \frac{1}{N} \sum_{i=1}^{N} \sum_{Z} \log p(y_i|Z, x_i, \theta) \prod_{l=1}^{L} p(Z_l|x_i, \theta, \phi) \\ \leq& \frac{1}{N} \sum_{i=1}^{n} \log \sum_{Z} p(y_i|Z, x_i, \theta) \prod_{l=1}^{L} p(Z_l|x_i, \theta, \phi). \end{aligned} \tag{3.2}$$

Instead of Equation (3.1), we take the first part of the Equation (3.2) in the CUTE training.

3.2. Routing

In the CUTE architecture, routing is managed by H units. H unit of layer L_i controls which of the parallel blocks of L_{i+1} will be used. H units consist of global average pooling layers and fully connected layers. Their computational cost is minimal compared to F units.

The use of Information Gain objectives to route samples to a subset of the network was proposed in Biçici et al. [4], where ground truth labels were used to compute the Information Gain. In this thesis, we propose Unsupervised Information Gain Loss. Alongside the classification objective, we also use the unsupervised information gain losses, IG_l, to route samples to computational units in every layer l.

With these two types of losses (classification and routing losses) and additional regularizers, global loss becomes

$$L_{CUTE} = -J_C(\theta, \phi) - \lambda_{IG} \sum_{l=1}^{L} IG_l + \Omega(\theta, \phi). \tag{3.3}$$

3.3. Unsupervised Local Information Gain Loss

The outputs of H units define a probability distribution $p(Z_l|z, \theta, \phi)$ over the F units of the next layer. The parameters of routing networks ϕ and the main network θ are trained with losses defined using the information gain calculated over these local probability distributions. We achieve a decision tree-like purification inside each parallel unit using these information gain-based loss functions. Decision trees mostly use greedy search in the feature space and find the optimal split that minimizes class impurity between sibling nodes. Even though popular decision tree algorithms such as ID3 [39] and C4.5 [40] use similar information gain maximization approaches, these mechanisms are not differentiable since they involve discrete operations such as counting data points to calculate information entropy. Bicici et al. [4] proposed a tree-structured neural network with differentiable information gain-based routing. This thesis introduces a trellis-shaped neural network with an unsupervised local information gain-based routing.

We define an unsupervised information gain loss similar to Bicici et al.'s work [4]. The main difference between ours and their loss definition is that we do not use the label information. We define the following joint probability distribution for each block $l > 0$ as

$$p(Z_l, x|\theta, \phi) = p(x)p(Z_l|x, \theta, \phi). \tag{3.4}$$

We define $p(Z_l|x, \theta, \phi)$ as a softmax over possible values of Z_l as

$$p(Z_l = k|x, \theta, \phi) = \frac{\exp\left(w_k^T h_l(x) + b_k\right)/\tau}{\sum_{i=0}^{K_l} exp(w_i^T h_l(x) + b_i)/\tau}. \tag{3.5}$$

The hyperplanes w_k, b_k determine the decision boundaries of the routers and $h_l(x)$ is the output of the transformations in the H units. Input x of the $h_l(x)$ is the output of the corresponding F_l unit. τ is a hyperparameter controlling the smoothness of the softmax output distribution. As $\tau \to \infty$, the distribution approaches to uniform and as $\tau \to 0$,

distribution approaches to onehot representation, setting the $(\texttt{argmax}_i(w_i^T h_l(x) + b_i))$ entry to one and the others to zero.

The unsupervised local information gain at layer l is defined as

$$IG_l = \mathbb{H}[p(Z_l)] - \mathbb{H}[p(Z_l|x, \theta, \phi)]. \tag{3.6}$$

Here $\mathbb{H}[p(x)]$ is defined as the entropy of the probability distribution as:

$$p(x) : \mathbb{H}[p(x)] = -\sum_x p(x) \log p(x). \tag{3.7}$$

The first term, $\mathbb{H}[p(Z_l)]$, acts as a regularizer term that prevents all samples from taking the same route. This term is calculated as the mean over the batch activations of the routing networks for each route. Since we are trying to minimize the information entropy for the routing activations, the network sends the same number of samples to each route. The second term is calculated from the activations of each sample. By minimizing $-\mathbb{H}[p(Z_l|x, \theta, \phi)]$, the network tries to increase the entropy of routing activations for each sample and make routing choices more confident.

3.4. Training With Information Gain Loss

The derivatives of the lower bound in Equation (3.2) with respect to the main network parameters θ are given as

$$\begin{aligned}\frac{\partial J_C}{\partial \theta} =& \frac{1}{N} \sum_{i=1}^{N} \sum_{Z} \frac{\partial \log p(y_i|Z, x_i, \theta)}{\partial \theta} \prod_{l=1}^{L} p(Z_l|x_i, \theta, \phi) \\ &+ \sum_{i=1}^{N} \sum_{Z} \log p(y_i|Z, x_i, \theta) \frac{\partial \prod_{l=1}^{L} p(Z_l|x_i, \theta, \phi)}{\partial \theta}\end{aligned} \tag{3.8}$$

and with respect to the routing network parameters ϕ are given as

$$\frac{\partial J_C}{\partial \phi} = \frac{1}{N} \sum_{i=1}^{N} \sum_{Z} \log p(y_i|Z, x_i, \theta) \frac{\partial \prod_{l=1}^{L} p(Z_l|x_i, \theta, \phi)}{\partial \phi}. \tag{3.9}$$

While the first term of $\frac{\partial J_C}{\partial \theta}$ leads to a simple Monte Carlo approximation by drawing samples $x_i \sim p(x)$ and $Z_i \sim p(Z|x_i, \theta, \phi)$; $\frac{\partial J_C}{\partial \phi}$ and the the second term of $\frac{\partial J_C}{\partial \theta}$ is not expectations with respect to a distribution. Therefore, Monte Carlo approximation is not an option. In general, the gradients of the type $\frac{\partial \mathbb{E}_{p(z|w)}[f(x)]}{\partial w}$ are difficult to compute. Hence, there are other solutions such as gumbel softmax [23, 24] to approximate such cases.

In CUTE, we completely ignore $\frac{\partial J_C}{\partial \phi}$, shown in Equation (3.9) and the second part of the $\frac{\partial J_C}{\partial \theta}$ from Equation (3.8), whose first part we will call $\frac{\partial \hat{J}_C}{\partial \theta}$. The routing network parameters θ are trained merely using the information gain losses at each block. The main network parameters θ are also trained using the information gain objectives.

While training the network, at each block l, the router unit uses the outputs of the computational units F to build and sample from probability distributions $p(Z_l|\hat{x_l}, \theta, \phi)$ where $\hat{x_l}$ are the outputs for each corresponding F unit. According to the samples Z_l, the corresponding F units in the next block are activated for the current sample x_i. In general, this is equivalent to sampling a root-to-leaf path for each sample, evaluating and backpropagating the corresponding error signal through the followed path. Since we only partially use the primary loss gradients and depend on the information gain losses to train the router networks, we call this algorithm the "Approximate Training". The pseudo-code for the algorithm is shown in Figure 3.4. We also let the routers use a uniform sampling for a certain number of epochs before enabling information gain objectives to train each route with every possible sample at the beginning. Our experiments show that this is an effective regularizer for the CUTE.

Only a single path is evaluated during the inference of sample x, making the algorithm more efficient than unconditional baselines. During inference, the most likely F unit is chosen from the corresponding H unit's output for each sample x. This expert path is the most appropriate because it is trained with samples semantically similar to x; hence, it has high discriminative power.

for $0 < epoch < N_{warmup}$ **do**

Use uniform sampling at the routers

Train the rest of the network like a conventional CNN

end for

for $N_{warmup} <= epoch < N_{training}$ **do**

Sample a minibatch $\{x_i, y_i\}_{i=1}^{B} \sim p(x, y)$

for $1 \leq i \leq B$ **do**

$\hat{x}_0^i = F_0(x_i)$

Sample next layer's F units: $Z_1^i \sim p(Z_1^i | \hat{x}_0^i, \theta, \phi)$

end for

Evaluate information gain: $IG_1 = \mathbb{H}[p(Z_1)] - \mathbb{H}[p(Z_1|x, \theta, \phi)]$

Compute gradients: $\dfrac{\partial IG_1}{\partial \theta}$ and $\dfrac{\partial IG_1}{\partial \phi}$

for $1 \leq l \leq L$ **do**

for $1 \leq i \leq B$ **do**

$j = \arg\max\{Z_l^i\}$

$\hat{x}_l^i = F_{l,j}(\hat{x}_{l-1}^i)$

Sample next layer's F units: $Z_{l+1}^i \sim p(Z_{1+1}^i | \hat{x}_l^i, \theta, \phi)$

end for

Evaluate information gain: $IG_{l+1} = \mathbb{H}[p(Z_{l+1})] - \mathbb{H}[p(Z_{l+1}|x, \theta, \phi)]$

Compute gradients: $\dfrac{\partial IG_l}{\partial \theta}$ and $\dfrac{\partial IG_l}{\partial \phi}$

end for

for $1 \leq i \leq B$ **do**

Evaluate posterior: $p(y_i | Z_i, \hat{x}_L^i, \theta)$

Compute gradient: $\dfrac{\partial \log p(y_i | Z_i, \hat{x}_L^i, \theta)}{\partial \theta}$

end for

end for

Figure 3.4. Approximate CUTE Training Algorithm.

4. EXPERIMENTS

CUTE (Conditional Unsupervised Information Gain Trellis) architecture is a conditional neural network structure designed for classification using conditional routing methods for the parallel computational units. Before each layer containing parallel computational units, CUTE has routing networks designed to route samples to the corresponding parallel unit of the following layer. Each sample activates one of these parallel computational units, and features are extracted for the next computational unit and routing network. These routing networks are trained using unsupervised information gain loss, aiming to cluster similar images and send them to similar paths, creating expert computational nodes that can better classify a subset of classes.

We experimented with and inspected the clustering performance of neural networks optimized with unsupervised information gain loss. We hypothesized that our unsupervised information gain loss clusters samples with similar features. To show the clustering performance of our unsupervised information gain loss, we designed clustering experiments with several datasets consisting of different data distributions and scenarios in Section 4.1.

In addition to clustering experiments with small neural networks, classification experiments have been performed on Fashion MNIST [7] using the CUTE version with a LeNet [41] architecture in Section 4.2. Alongside the classification performance of the CUTE networks, we have inspected the routing behaviors of the CUTE architecture.

4.1. Unsupervised Information Gain Clustering Experiments

Unsupervised information gain loss is designed explicitly to route inputs with similar features to similar paths to create a mixture of experts classifier that can classify a subset of the data better than a standard network trained with all of the training data. Although the CUTE network is trained using a supervised classification loss,

routing is based on an unsupervised information gain loss.

The experiments on Fashion MNIST have shown that given input with good representational power, unsupervised information gain objective does a decent job routing similar inputs to similar paths. However, we wanted to show that the unsupervised information gain objective can also be a standalone objective function for the clustering task. The clustering results are pretty similar to the well-known K-Means [42] algorithm regarding performance and clustering behavior.

4.1.1. Method

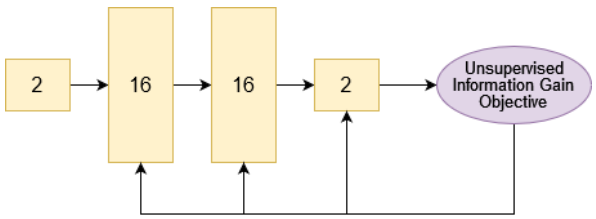

Figure 4.1. The small fully connected neural network used for clustering experiments.

In clustering experiments, a small, fully connected neural network with two inputs, two hidden layers having 16 nodes, and two nodes have been employed at the output layer, as shown in Figure 4.1. Networks have been trained with the input data for ten epochs using the stochastic gradient descent optimizer with a 0.05 learning rate. This fully connected network can be thought of as a routing network in a CUTE architecture. In these experiments, we try to understand the clustering performance of these routing networks trained separately from a classification network.

4.1.2. Datasets

In clustering experiments, various custom two-dimensional datasets are used as seen in Figure 4.2 with the expected cluster results to visualize different clustering scenarios and compare the results visually.

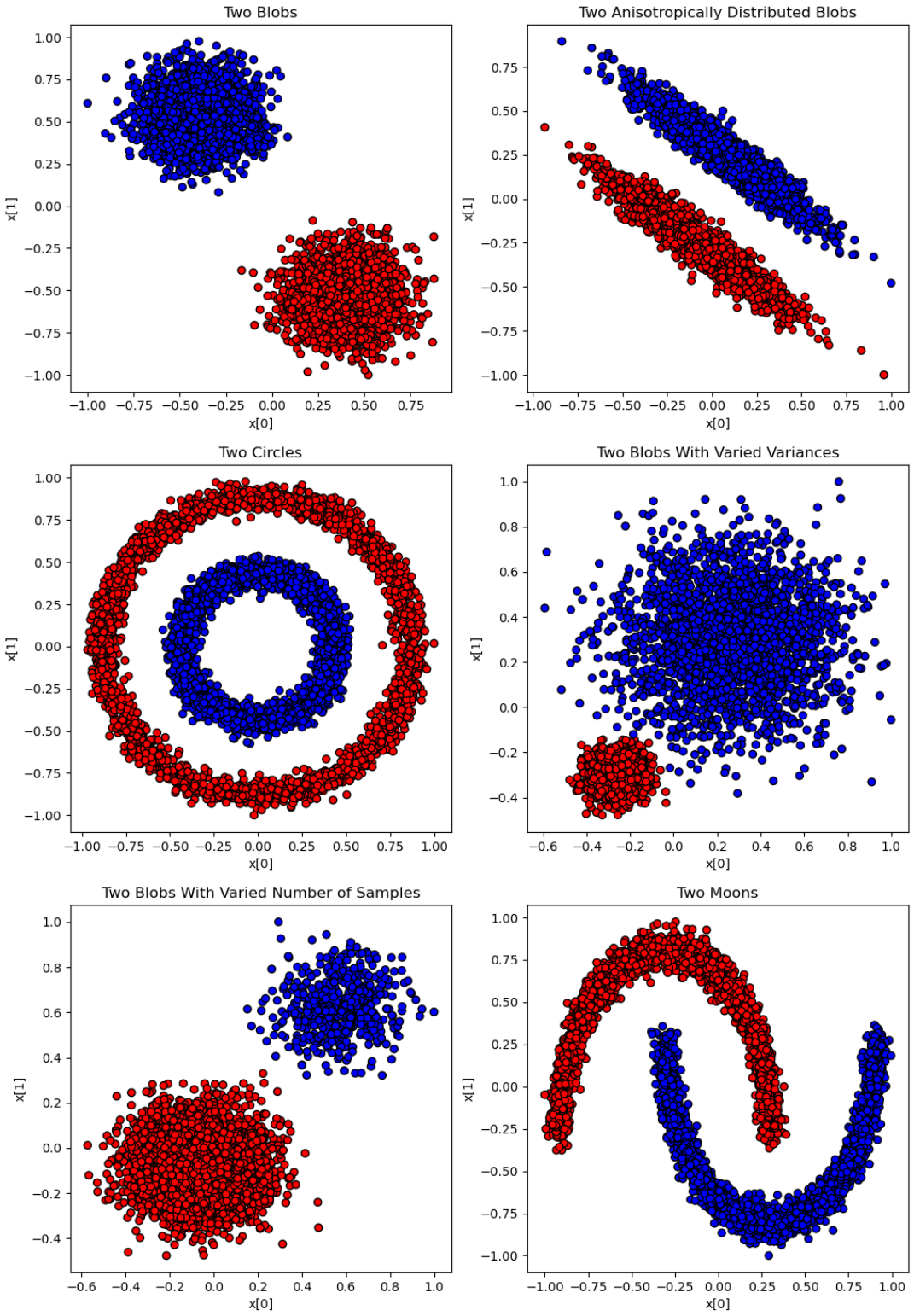

Figure 4.2. Datasets populated for clustering experiments.

The first dataset contains two normally distributed data blobs with different centers and the same variance and number of samples. The second is a dataset consisting of two anisotropically distributed data blobs with different centers and the same variance and number of samples. The third is a dataset consisting of a large circle containing a smaller circle in two dimensions with the same number of samples. Next, a dataset that contains two normally distributed blobs with the same number of samples and different centers and variances has been employed. In addition, a dataset with two normally distributed blobs with different centers and sample sizes and the same variances has been used. Lastly, the two moons dataset consisting of two interleaving half circle-shaped blobs with the same number of samples has been employed.

4.1.3. Performance Metrics

To compare the results of the clustering neural network with K-Means, we have inspected normalized mutual information, and the Rand Index [43] for the clustering results. In most scenarios, the results were similar, although unsupervised information gain clustering has the upper hand for some scenarios compared to K-Means.

Normalized mutual information (NMI) gain is a normalization of mutual information, also called information gain, scored on a scale between 0 and 1. NMI score 0 means no mutual information, while NMI score 1 means total correlation. Assuming two label distributions of the same dataset, U (ground truth labels) and V (predicted cluster labels); the entropy of the label distributions are calculated using Equation (3.7) as

$$\mathbb{H}[U] = -\sum_{i=1}^{|U|} p(i) \log(p(i)) \tag{4.1}$$

and

$$\mathbb{H}[V] = -\sum_{j=1}^{|V|} p(j) \log(p(j)) \tag{4.2}$$

where $p(i) = |U_i|/N$ is the probability that an object picked at random from U belongs to class U_i, and $p(j) = |V_j|/N$ is the probability that an object picked at random from V belongs to class V_j. The mutual information between U and V is calculated by

$$\mathbb{MII}[U, V] = \sum_{i=1}^{|U|} \sum_{j=1}^{|V|} p(i, j) \log \left(\frac{p(i, j)}{p(i)p(j)} \right). \tag{4.3}$$

The normalized mutual information is, then, defined as

$$\mathbb{NMII}[U, V] = \frac{\mathbb{MII}(U, V)}{\mu(\mathbb{H}[U], \mathbb{H}[V])}. \tag{4.4}$$

Rand index calculates the similarity of the two assignments, disregarding permutations. It can be viewed as a binary classification over the pairs of elements in the given dataset. Assuming two label distributions of the same dataset, U (ground truth labels) and V (predicted cluster labels); to define Rand index, we need to define the two following variables a and b as

$$a = |\{(o_i, o_j) | o_i, o_j \in U_k, o_i, o_j \in V_l\}| \tag{4.5}$$

and

$$b = |\{(o_i, o_j) | o_i \in U_k, o_j \in U_m, k \neq m, o_i \in V_l, o_j \in V_n, l \neq n\}|. \tag{4.6}$$

Variable a represents the number of pairs of elements belonging to the same clusters in U and V, while variable b is the number of pairs belonging to the different clusters in U and V.

Therefore, the Rand index (RI) is calculated as

$$\mathbb{RI} = \frac{a + b}{n(n-1)/2}. \tag{4.7}$$

4.1.4. Experiments

In the first experiment, both k-means and unsupervised information gain clustering performed similarly, as seen in Figure 4.3 and Table 4.1. An attractive property is observed when the logit values obtained from unsupervised information gain clustering output are visualized. Logit outputs are similar to a PCA projection of the input data into a line, making it easier to distinguish samples.

For each dataset, we have visualized the training data distribution on the upper left sides of the figures; K-Means clustering results on the upper right side of the figure; clustering results of the neural network trained with unsupervised information gain clustering (UIGC) on the lower right sides of figures; logit outputs of the neural network colored with the ground truth labels on the lower left sides of the figures.

Unsupervised information gain clustering performed better in the second experiment, as seen in Figure 4.4 and Table 4.2. We have seen the same property of logit values as in the first experiment. With these results, we can conclude that our method is more robust to irregular manifolds and anisotropic distributions than K-Means clustering.

In the third experiment, unsupervised information gain clustering performed as poorly as K-Means clustering, as seen in Figure 4.5 and Table 4.3. We have seen the same property of logit values as in the first experiment. With these results, we can conclude that our method is not robust to a dataset that is not linearly separable.

In the fourth experiment, unsupervised information gain clustering performed better at both metrics than K-Means clustering, as seen in Figure 4.6 and Table 4.4. We have seen the same property of logit values as in the previous experiments. We can conclude that our method is more robust to variance difference than K-Means clustering with these results.

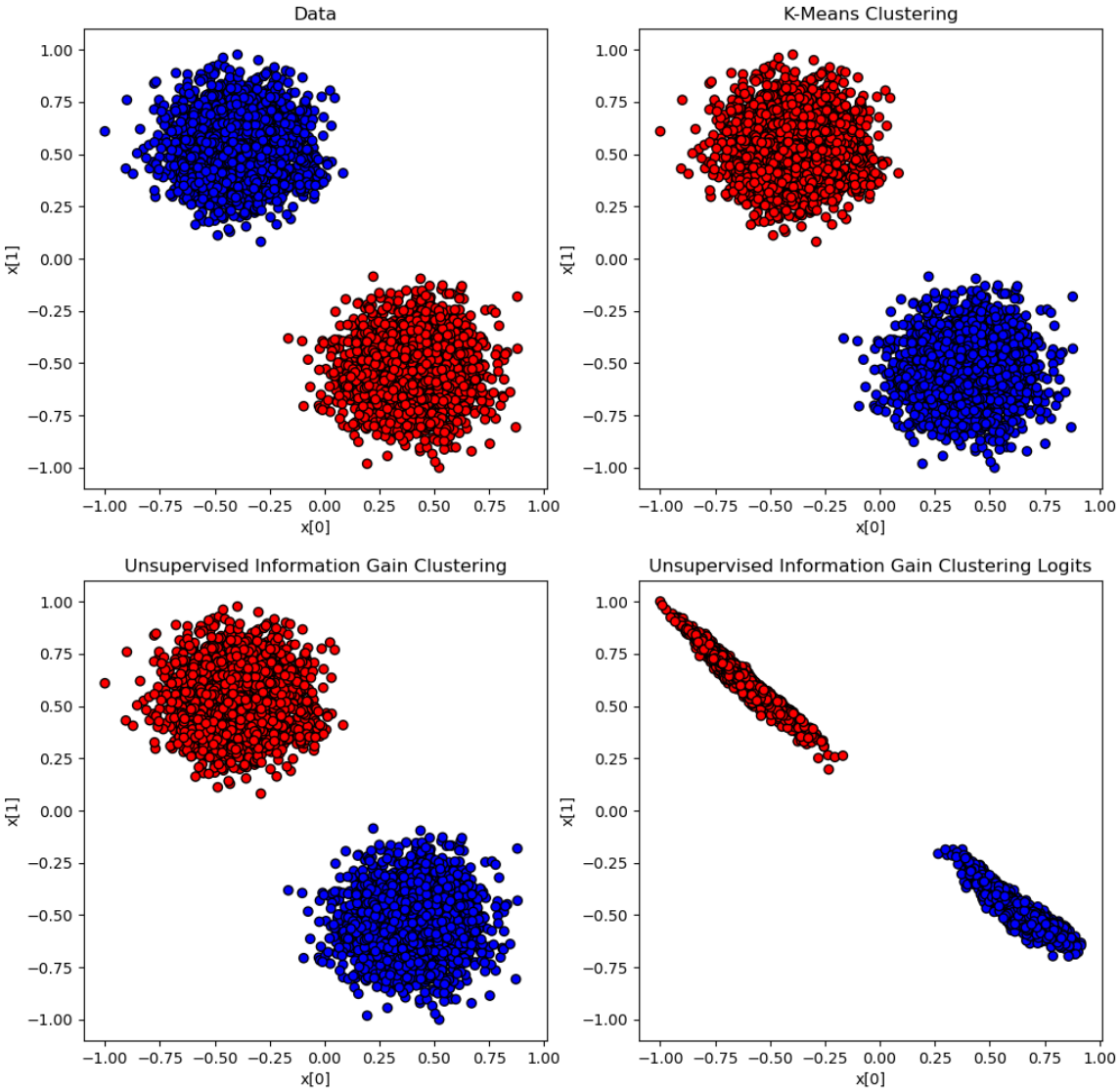

Figure 4.3. Experiment 1: Clustering results on two normally distributed data blobs with different centers and same variance and number of samples.

Table 4.1. Experiment 1: Clustering scores of the normally distributed data blobs with different centers and same variance and number of samples.

	NMI Score	**Rand Index**
K-Means	1.0	100.00%
UIGC (Ours)	1.0	100.00%

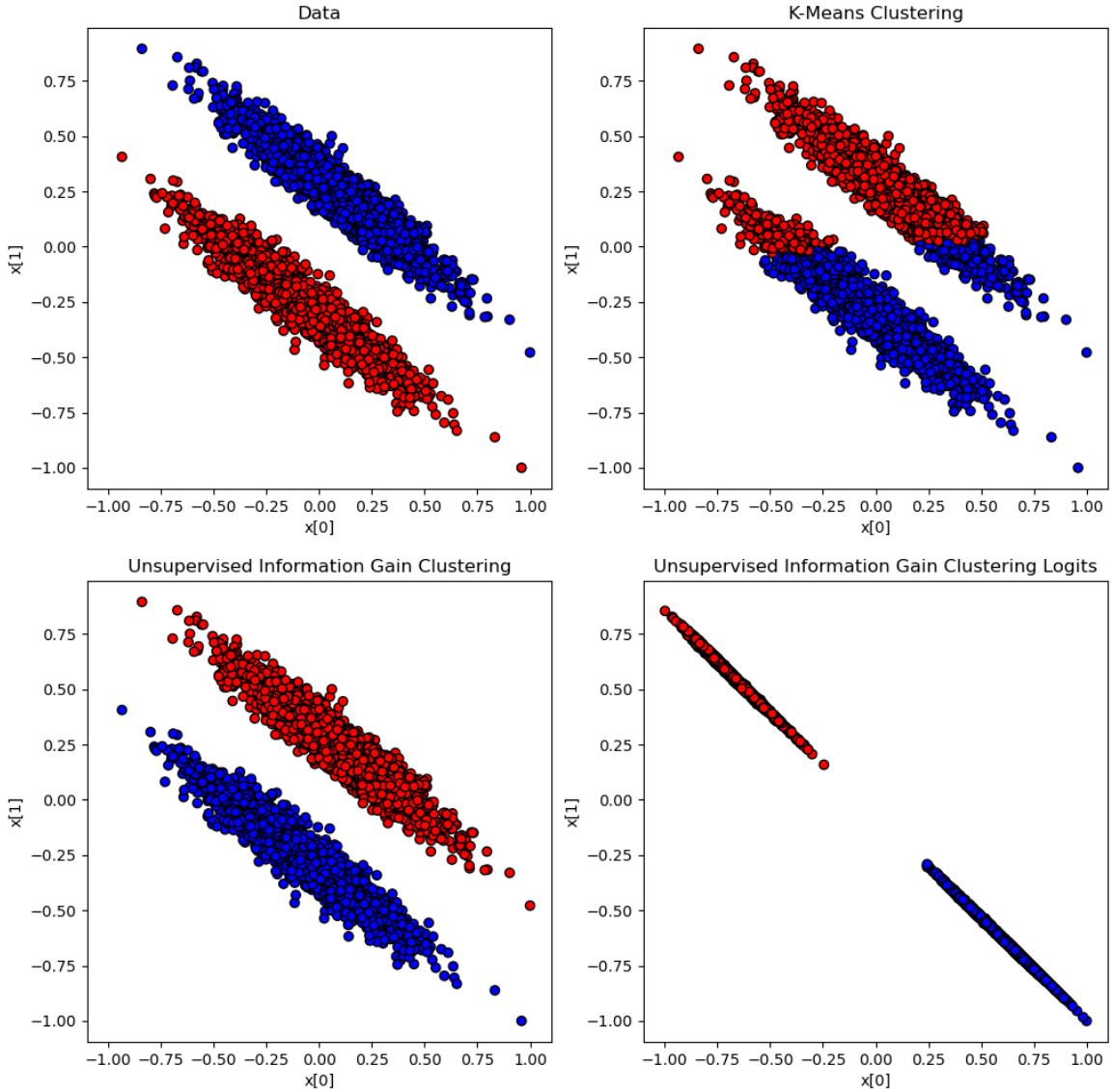

Figure 4.4. Experiment 2: Clustering results on two anisotropically distributed data blobs with different centers and same variance and number of samples.

Table 4.2. Experiment 2: Clustering scores of the anisotropically distributed data blobs with different centers and same variance and number of samples.

	NMI Score	**Rand Index**
K-Means	0.5	80.44%
UIGC (Ours)	1.0	100.00%

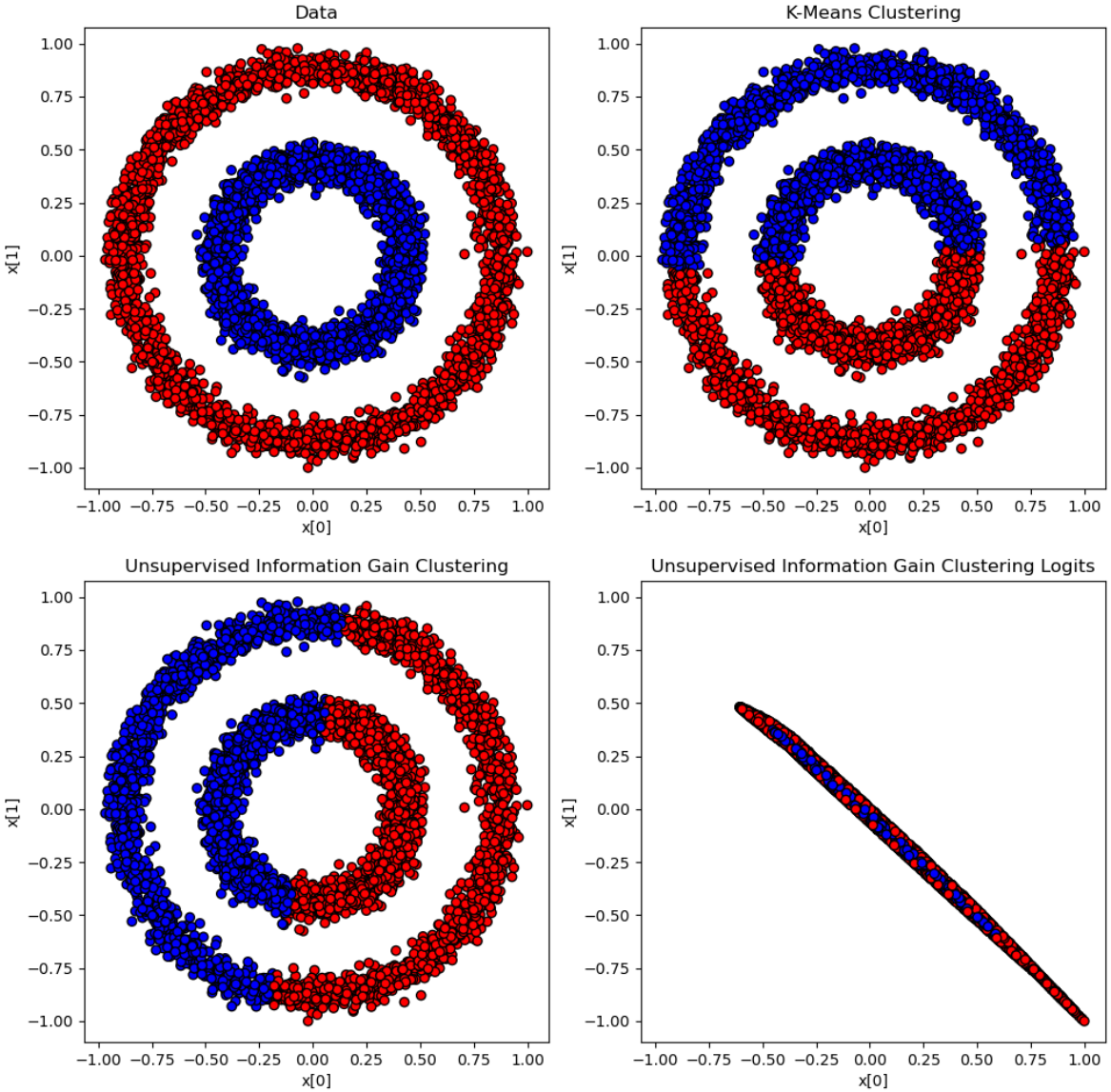

Figure 4.5. Experiment 3: Clustering results on a Large circle containing a smaller circle with the same number of samples.

Table 4.3. Experiment 3: Clustering scores of dataset with concentric circles.

	NMI Score	**Rand Index**
K-Means	0.0	49.99%
UIGC (Ours)	0.0	49.99%

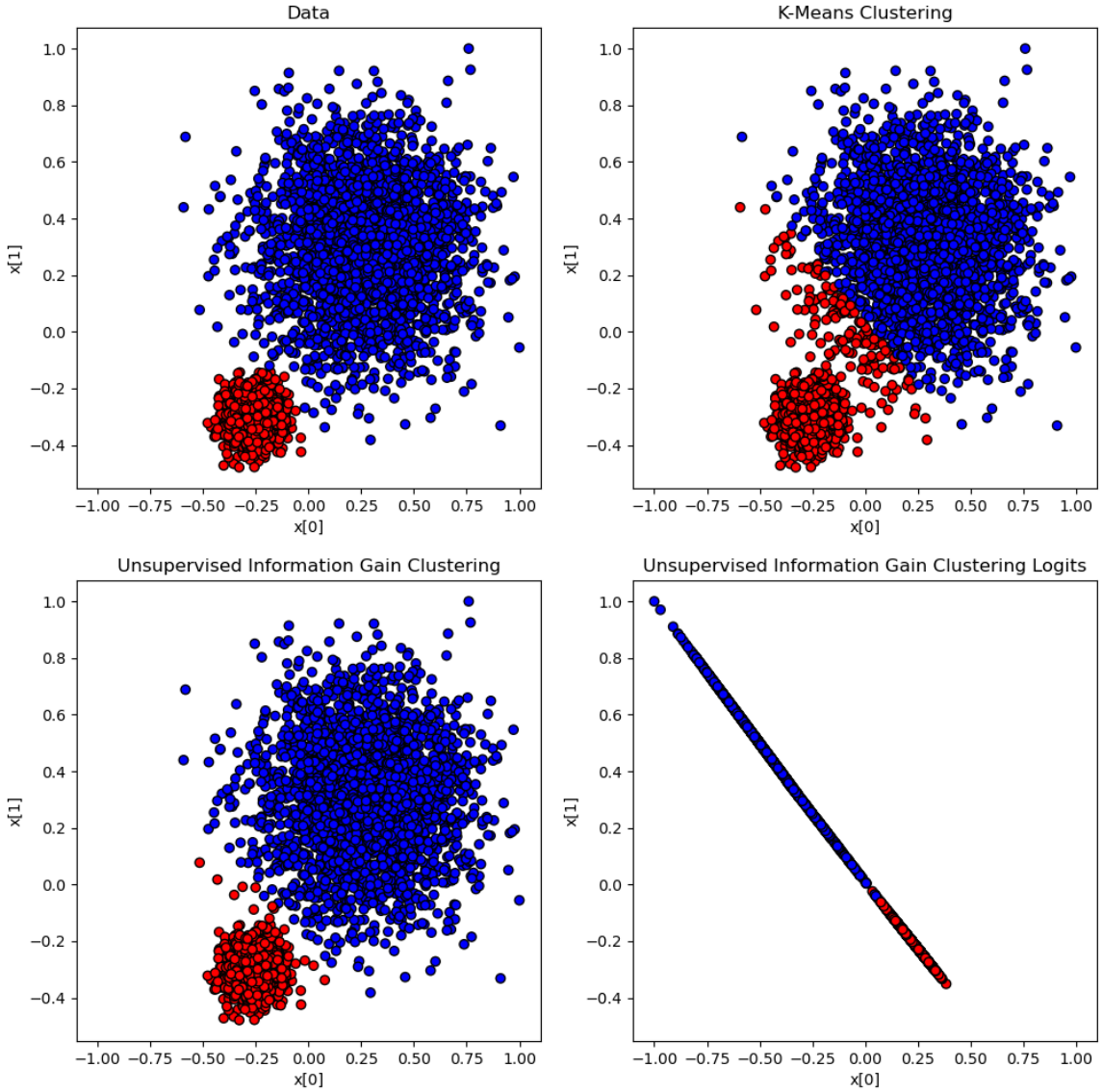

Figure 4.6. Experiment 4: Clustering results on two normally distributed data blobs with different centers and variances, and same number of samples.

Table 4.4. Experiment 4: Clustering scores of the normally distributed data blobs with different centers and variances, and the same number of samples.

	NMI Score	**Rand Index**
K-Means	0.83	93.88%
UIGC (Ours)	0.98	99.52%

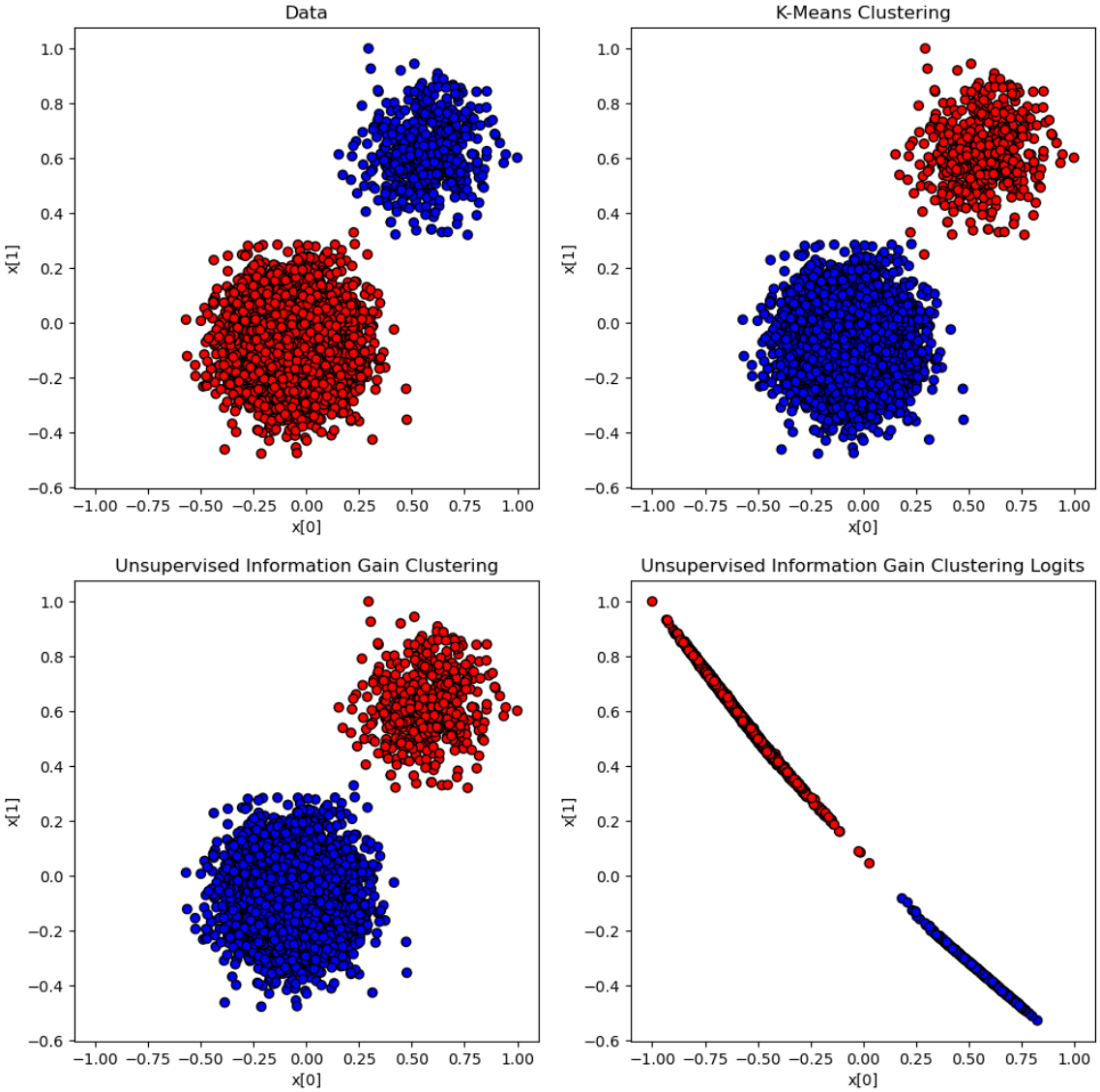

Figure 4.7. Experiment 5: Clustering results on two normally distributed data blobs with different centers and number of samples, and same variances.

Table 4.5. Experiment 5: Clustering scores of the normally distributed data blobs with different centers and number of samples, and same variances.

	NMI Score	**Rand Index**
K-Means	0.99	99.92%
UIGC (Ours)	1.0	100.00%

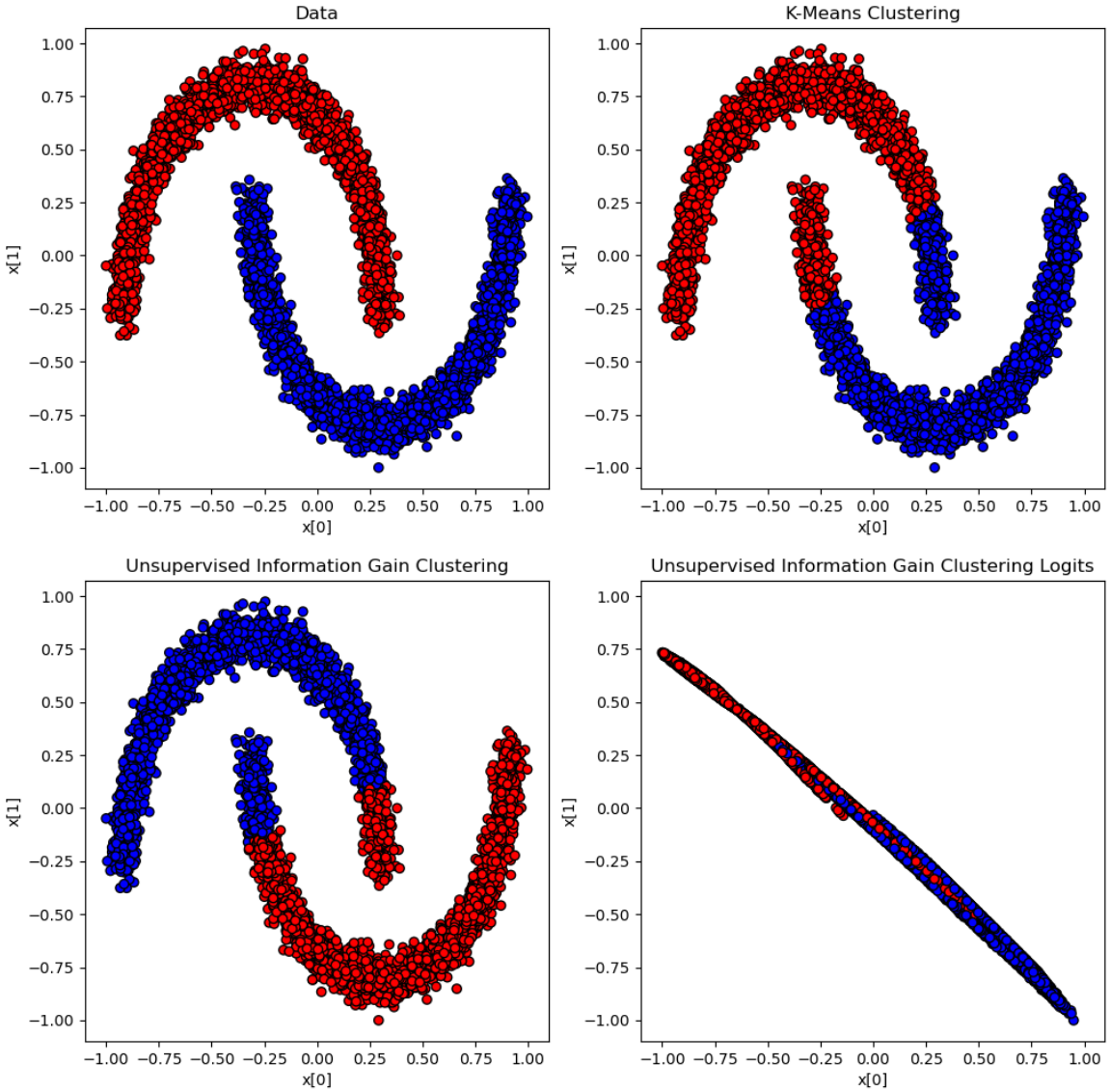

Figure 4.8. Experiment 6: Clustering results on two moons dataset.

Table 4.6. Experiment 6: Clustering scores of the two moons dataset.

	NMI Score	**Rand Index**
K-Means	0.39	74.41%
UIGC (Ours)	0.48	79.49%

In the fifth experiment, unsupervised information gain clustering performed similarly at both metrics with K-Means clustering, as seen in Figure 4.7 and Table 4.5. We have seen the same linear property of logit values as in the previous experiments. With these results, we can conclude that our method is as robust to data imbalance as K-Means clustering. If class imbalance would be a problem for unsupervised clustering, we can overcome this issue by introducing a weight parameter to Equation (3.6) and modifying the equation as

$$IG_{l_{balanced}} = \lambda_{balance}\mathbb{H}[p(Z_l)] - \mathbb{H}[p(Z_l|x,\theta,\phi)]. \tag{4.8}$$

In the last experiment, unsupervised information gain clustering performed poorly at both metrics with K-Means clustering, as seen in Figure 4.8 and Table 4.6. We have seen the same linear property of logit values as in the previous experiments. With these results, we can conclude that our method is not robust to data that is not linearly separable, as seen in the third clustering experiment.

4.2. CUTE Classification Experiments on the Fashion MNIST dataset

CUTE is designed as an image classification model based on a conditional artificial neural network structure. In this section, we have investigated the classification performance of the CUTE architecture using a CUTE version of LeNet architecture on the Fashion MNIST dataset, alongside the routing behaviors of routing networks and different routing behaviors of each class.

4.2.1. Dataset

Fashion MNIST [7] dataset replaces the well-known MNIST [8] and has similar properties such as image size (28×28 gray-scale images) and the sizes of training and test sets (60000 and 10000 images, respectively). It also contains ten classes containing different types of clothing: t-shirt, trouser, pullover, dress, coat, sandal, shirt, sneaker,

bag, and ankle boot. The fashion MNIST dataset is designed to share the same image size and structure of training and test splits with the MNIST dataset but is considered more complex than MNIST.

Figure 4.9. Example Images From Fashion MNIST Dataset. Starting from upper left: T-Shirt/Top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag and Ankle boot.

In the Fashion MNIST dataset, there are 60000 training images and 10000 test images with balanced class labels. While optimizing CUTE hyper-parameters, we have split 10% of the training images for validation and applied grid search over the network hyper-parameters, learning rate, number of warm-up epochs, dropout rate and weight parameters introduced in loss functions. Example images from each class are shown in Figure 4.9

Multi-class clusters in the Fashion MNIST dataset utilize our routing mechanisms as high-level classifiers. When we inspected the t-SNE embeddings of the Fashion MNIST training set images, shown in Figure 4.10, some clusters of embeddings, such as trousers, are easily distinguished from others. Additionally, sandals, sneakers, and ankle boots are clustered close to each other. The rest of the classes are also clustered within same-class images, although some class embeddings are intertwined, such as pullovers, coats, and shirts. With mixtures of experts approach of CUTE, we aim to create classifiers that are better focused on a subset of classes in the dataset.

Figure 4.10. t-SNE Embeddings of Fashion MNIST Training Images.

4.2.2. Architecture

We have used a convolutional neural network (CNN), shown in Figure 4.11, consisting of 2 convolutional and three fully connected layers as our baseline. The convolutional layers consist of 32 and 64 convolutional filters, respectively, with a filter size of 5×5. Dense layer sizes are 1024, 512, and 10, respectively. The conversion from baseline architecture into CUTE architecture is carried out using the procedure depicted in Figure 3.3.

Figure 4.11. The Baseline Architecture used in Fashion MNIST experiments.

We have created a network for the CUTE experiments, shown in Figure 4.12, with two routing units in the first router and two in the second router. Therefore, a CUTE route consists of 2 convolutional layers with 32, 32 filters. They are followed by ReLU activation and max-pooling layers of size 2×2. There are fully connected layers with 512, 512, and 10. The slim version of the CNN, shown in Figure 4.13, has the same architecture as one route of the CUTE network.

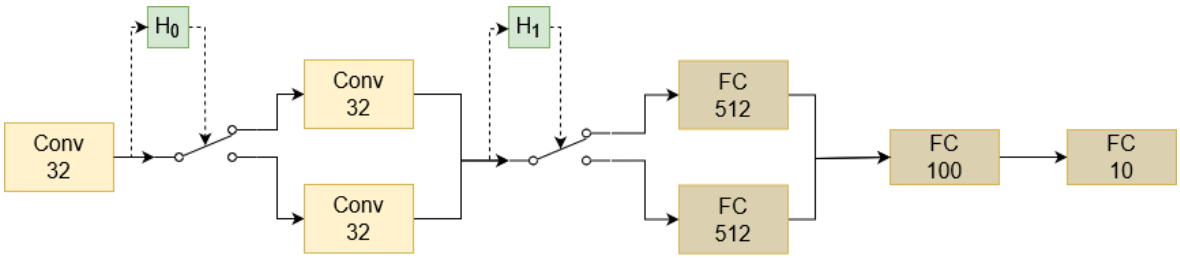

Figure 4.12. The CUTE Architecture used in Fashion MNIST experiments.

Figure 4.13. The Slim Baseline Architecture used in Fashion MNIST experiments.

We have used the Adam [44] optimizer. We have trained each model in 100 epochs with a batch size of 100 and a starting learning rate of 0.001. The learning rate starts with 0.001, is halved after 30^{th} and 60^{th} epochs ,and is divided by 10 in the 90^{th} epochs. After each fully connected layer in the convolutional and routing networks, we have used dropout regularization. We have conducted a grid search for the dropout rate within the interval $[0, 0.5]$ with the step size of 0.05. We have decided the optimal

dropout rate for the baseline and CUTE as 0.45 and 0.5, respectively. We have also conducted experiments on the slim baseline version, which resulted in lower accuracy. Random routing results were also inferior to our CUTE results, which means that our unsupervised routing method is not producing random results. We have used random routing at the start of each training to make each path more generalized and robust to routing errors. No routing loss is calculated while applying random routing. The number of steps for the random routing is also optimized using grid search within the interval [0, 30] with a grid size of 10 epochs. The value for the optimal random training epochs is set to 10.

4.2.3. Results

The results are shown in Table 4.7. Each experiment result is the test set accuracy of 5 runs with the best hyper-parameter configuration obtained from experiments performed on the validation set. Results show that our method can perform better than the slim baseline, with an equivalent number of parameters, but it needs more regularization to catch up with the baseline network. However, our method seems promising given the number of parameters and MAC operations.

Table 4.7. Fashion MNIST Results. Each sample visits a network equivalent to CNN (Slim) plus router blocks in CIGN and CUTE. CNN (R*) network uses CUTE architecture, but routing is handled randomly. Average # MAC column shows the average number of multiply-accumulate operations.

Method	Avg Acc.	Avg # MAC
CNN Baseline	92.65%	14.40×10^6
CNN (Slim)	92.21%	6.71×10^6
CNN (R*)	91.98%	6.71×10^6
CIGN [4]	92.37%	7.21×10^6
CUTE	**92.51%**	$\mathbf{7.21 \times 10^6}$

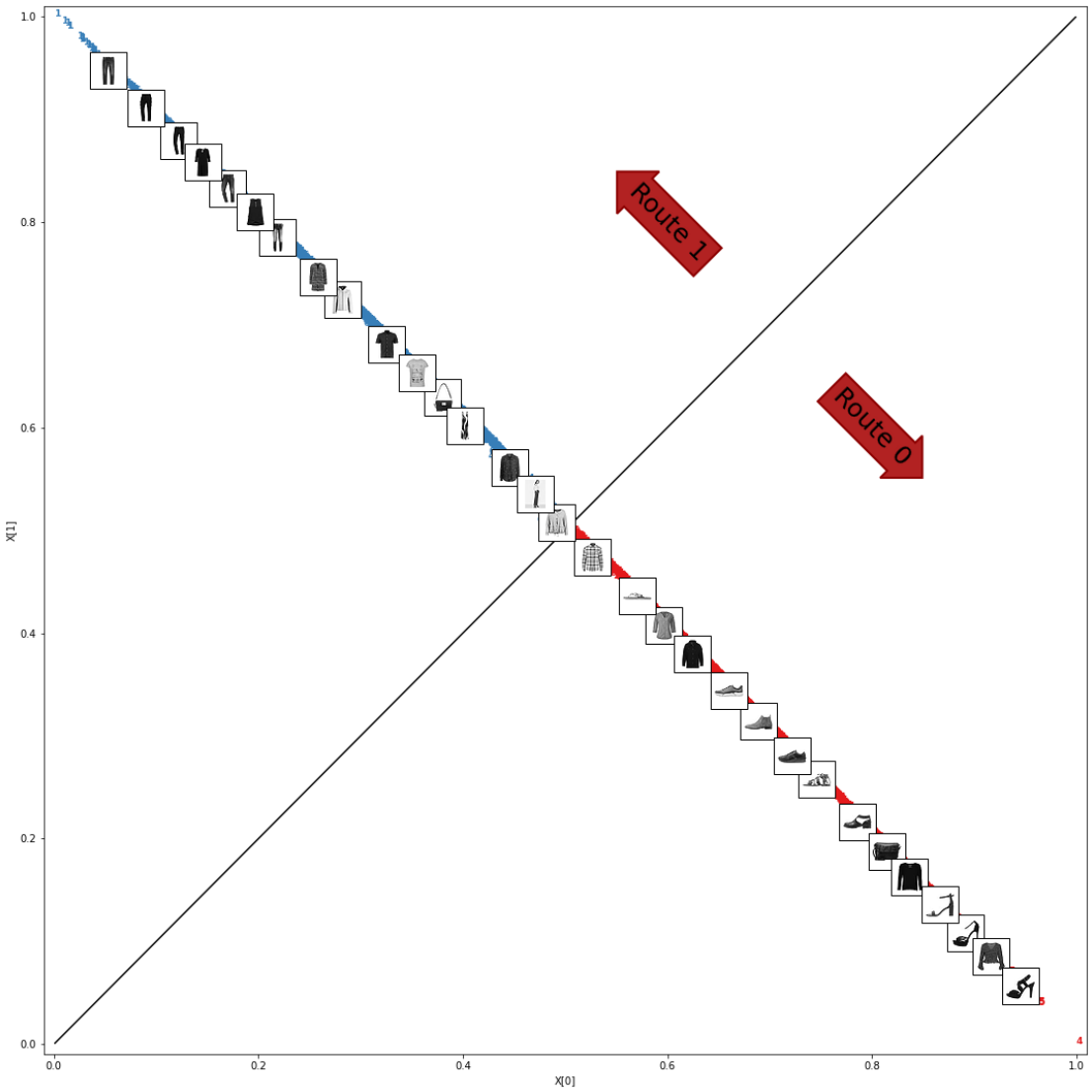

Figure 4.14. Visualization of Routing Logits for the First Routing Network.

In CUTE experiments, routing outputs have shown that our routing layers transform the input values fed from the previous layer, similar to dimensionality reduction. The transformation can be seen from the visualization of output values of routing networks. Since we have used two routes in each experiment, we used the output logit values directly in Figures 4.14 and 4.15. When logits from the routing network outputs are investigated, the embeddings lay over the line between points $(0, 1)$ and $(1, 0)$. As observed from the images, images are sorted according to their classes as trouser, dress, t-shirt, shirt, coat, pullover, sneaker, bag, sandal, and ankle boot. Trouser, dress, t-shirt, shirt, and coat follow the first path in the first routing layer, while pullover,

sneaker, bag, sandal, and ankle boot classes follow the second route. Similar routings are observed in the second routing network where ankle boot, sandal, sneaker, and pullover follow the first route while the rest follow the second route with a different ordering of classes alongside the embedding line.

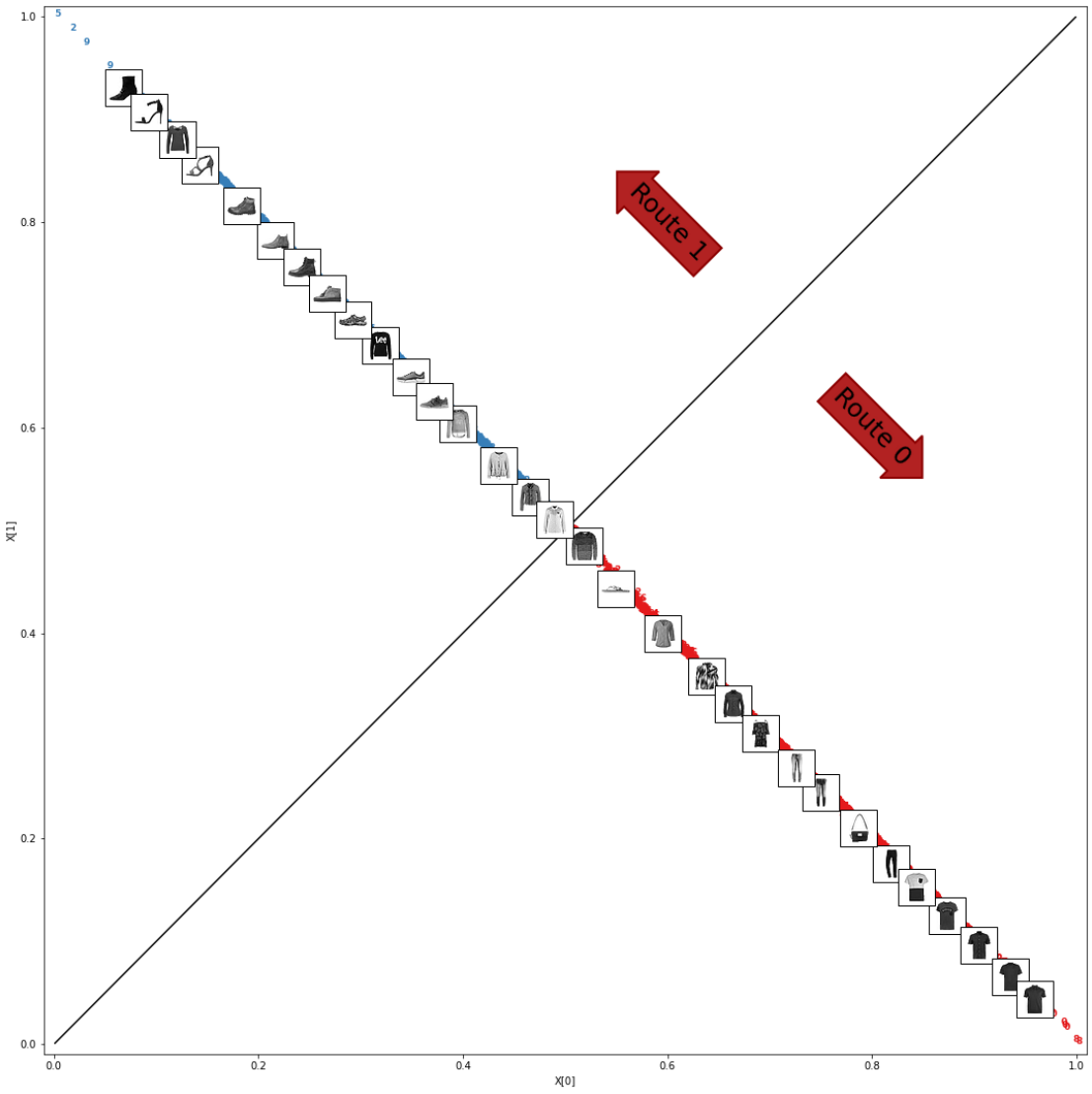

Figure 4.15. Visualization of Routing Logits for the Second Routing Network.

Investigating the routing behavior of the CUTE showed that similar classes tend to follow similar paths. Therefore, we can conclude that even though we do not use class supervision in our routing loss, it is possible to learn a routing mechanism to train a mixture of experts classifier for subsets of classes that are difficult to distinguish.

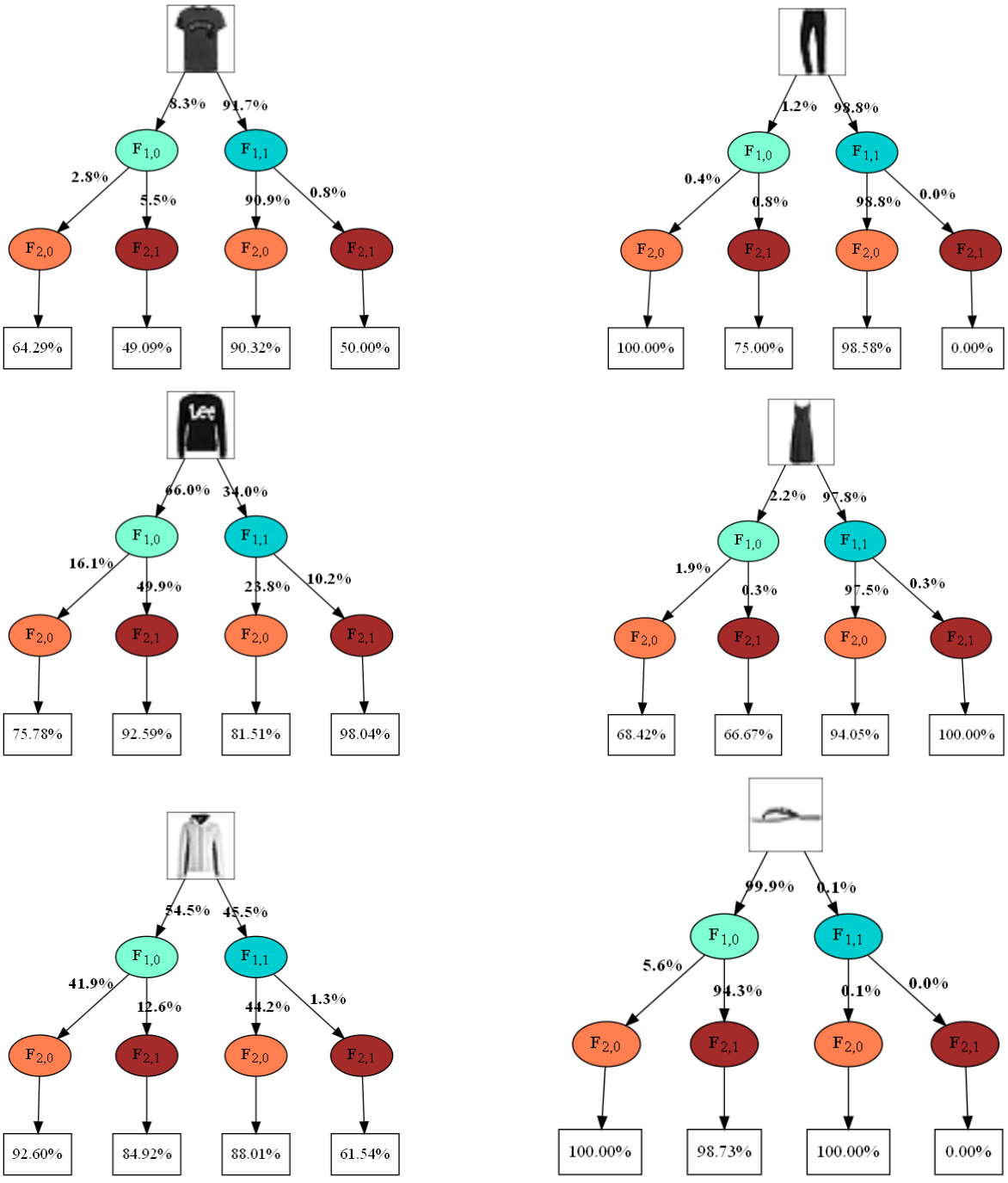

Figure 4.16. Routing statistics for T-Shirt/Top, Trouser, Pullover, Dress and Coat classes: Percentage of images in the test set that follow a particular class route are shown on the tree edges. The second-degree nodes (F_2) of the trellis structure are shown twice to make the results more readable. The classification accuracy of given classes following the root is shown at the leaves.

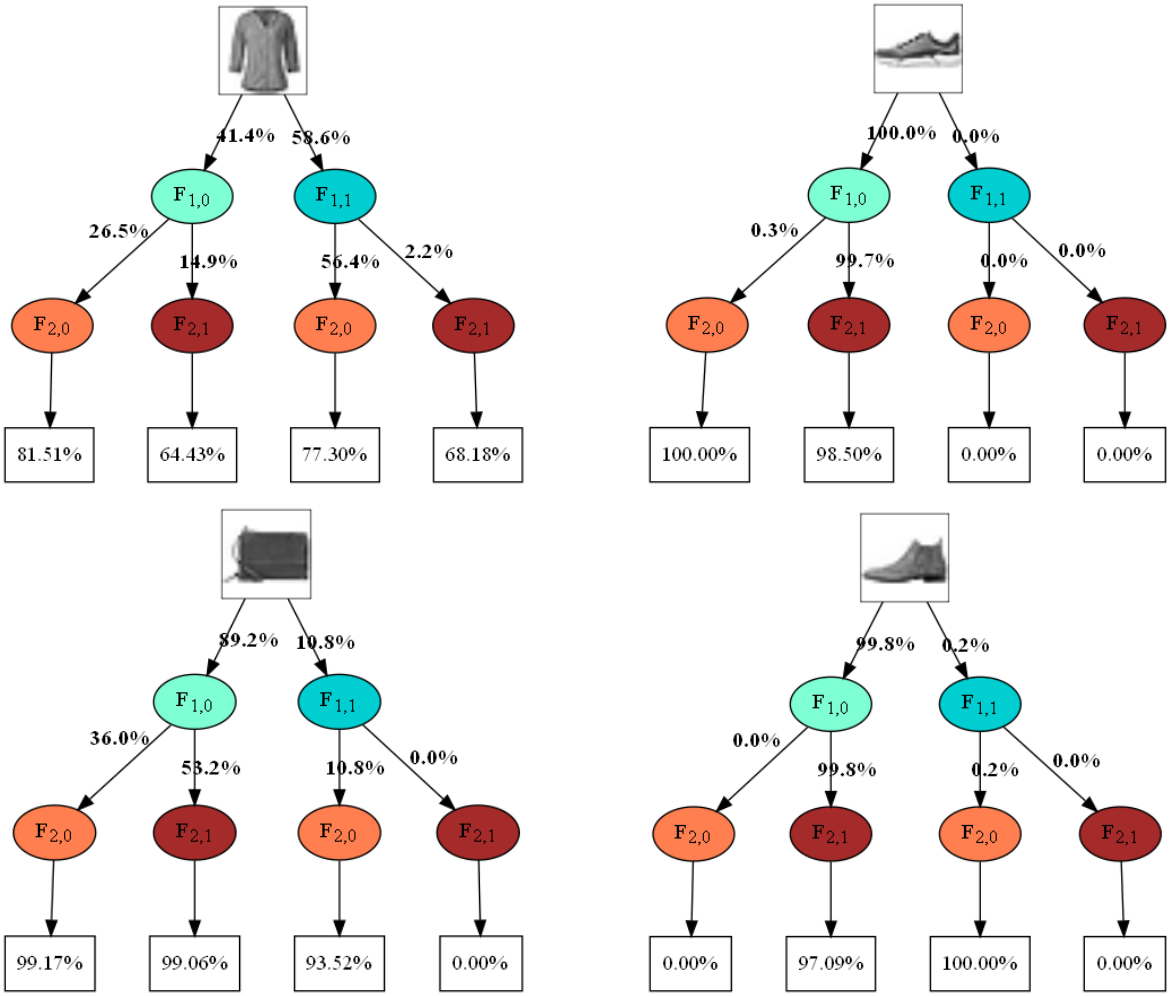

Figure 4.17. Routing statistics for Shirt, Sneaker, Bag and Ankle Boot classes: Percentage of images in the test set that follow a particular class route are shown on the tree edges. The second-degree nodes (F_2) of the trellis structure are shown twice to make the results more readable. The classification accuracy of given classes following the root is shown at the leaves.

The most apparent drawback of the CUTE architecture is that misrouting a sample may cause poor classification performance due to sub-optimal feature extraction. As seen in Figure 4.17, each class has a path which most of its samples follow, and a misrouted sample is more prone to misclassification. Compared to CIGN, which has a tree-structured network, our network architecture allows the second routing layer to compensate for the misrouting of the first routing network. T-Shirt/Top, Pullover and Shirt samples following the non-frequent route after the first routing classified better

when they routed to the most frequent route in the second routing classified better. For the T-Shirt/Top class, 2.8% of the samples are routed to the less frequent first route in the first routing but are routed to the most frequent first route in the second routing and classified better than those misrouted in the second routing. Therefore, we can conclude that our trellis structure has the advantage of compensating routing errors over the tree structure.

5. CONCLUSION

In this thesis, we proposed a trellis-shaped artificial neural network model, called Conditional Unsupervised Information Gain Trellis, CUTE, in which samples can be routed during training and inference stages, using routing networks trained with information gain-based unsupervised objectives. We have proposed methods and strategies for its training and inference. We have conducted clustering experiments using a network trained with our novel unsupervised information gain-based objective function. Moreover, we have experimented with our CUTE architecture against the well-known Fashion MNIST dataset. We show that CUTE performs on par with unconditional baselines with a heavier computational burden. It also performs favorably compared to similar conditional computation methods in terms of accuracy while achieving a comparable reduction in the number of used Multiply-And-Accumulate operations. We analyzed the routing behavior of our routing networks at each layer. Our investigations over the routing at different layers have shown that the trellis structure enables misrouted samples to recover. Additionally, we observed that semantically similar classes followed similar paths that specialized in classifying a subset of classes.

Deep learning models have been highly successful in solving the image classification challenge. In general, the better deep neural networks' classification performance, the more computationally expensive architectures become. Most prominent deep learning models handle inference in a static way, meaning that once training is completed, the network architecture and weight parameters are fixed.

Conditional computing generates new neural network architectures that can activate a subset of the network based on the input sample. As a result, they can dynamically limit the use of computational resources. Additional to efficiency advantage, conditional methods can significantly enlarge the parameter space, therefore, the representation power of the model. Computational architectures allow the user to trade-off accuracy and efficiency for the low processing power environments. Therefore, they

are more adaptable to different hardware platforms, especially embedding systems or mobile devices.

Although artificial neural networks are inspired by the neural system of the human, the most common neural architectures are sequential as opposed to research on the human brain that claims that the brain processes information in a dynamic way. However, conditional architectures allow networks to process input dynamically. In addition, dynamic conditional models allow us to observe which input parts are accountable for the network's predictions.

CUTE proposed two novel approaches to the conditional network domain. Firstly, the unsupervised information gain-based objective is introduced for the routing networks. Shortly, our objective function tries to increase the entropy of the routing predictions made by the router network while also preventing overfitting on a single route. Secondly, the trellis-shaped architecture is inspired by the tree-structured conditional computing methods but handles the disadvantage of misrouting in early layers by allowing it to be routed correctly at the following router nodes.

The future work for our model will be focused on regularizing the CUTE training scheme. We have observed that one of the main reasons for classification error is misrouting the samples from the class' most preferred path. We observed that the subset of classes split in the first router is split with a very similar distribution in the second router, even though a CUTE network with two routing networks with two possible routes creates four possible paths for an input sample. Therefore, most samples follow two paths out of four possible route combinations. Further experiments can be done with router networks with more possible routes, or a global objective function can be introduced to incentivize samples to use less-preferred routes. Another approach for increasing the classification performance of the CUTE networks can be applying soft gating mechanisms instead of hard gating at the routers. Soft gating allows us to utilize feature maps from multiple paths for the problematic samples.

We designed the CUTE network as a conditional dynamic deep learning architecture. However, we decide the network architecture beforehand and then train the network to adapt to the given architecture. Alternatively, another future work direction can be sequentially designing the network hierarchy by observing the data distribution of the routing networks.

REFERENCES

1. Krizhevsky, A., I. Sutskever and G. E. Hinton, "Imagenet Classification with Deep Convolutional Neural Networks", *Advances in Neural Information Processing Systems*, Vol. 25, pp. 1097–1105, 2012.

2. Deng, J., W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, "Imagenet: A Large-scale Hierarchical Image Database", *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, IEEE, 2009.

3. Hubel, D. H. and T. N. Wiesel, "Receptive Fields, Binocular Interaction and Functional Architecture in the Cat's Visual Cortex", *The Journal of Physiology*, Vol. 160, No. 1, pp. 106–154, 1962.

4. Biçici, U. C., C. Keskin and L. Akarun, "Conditional Information Gain Networks", *2018 24th International Conference on Pattern Recognition*, pp. 1390–1395, IEEE Computer Society, 2018.

5. Jordan, M. I. and R. A. Jacobs, "Hierarchical Mixtures of Experts and the EM Algorithm", *Neural Computation*, Vol. 6, No. 2, pp. 181–214, 1994.

6. Montillo, A., J. Tu, J. Shotton, J. Winn, J. E. Iglesias, D. N. Metaxas and A. Criminisi, "Entanglement and Differentiable Information Gain Maximization", *Decision Forests for Computer Vision and Medical Image Analysis*, pp. 273–293, Springer, 2013.

7. Xiao, H., K. Rasul and R. Vollgraf, "Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms", *arXiv Preprint arXiv:1708.07747*, 2017.

8. LeCun, Y., 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.

9. Hinton, G. E., S. Osindero and Y.-W. Teh, "A Fast Learning Algorithm for Deep Belief Nets", *Neural Computation*, Vol. 18, No. 7, pp. 1527–1554, 2006.

10. Raina, R., A. Madhavan and A. Y. Ng, "Large-scale Deep Unsupervised Learning Using Graphics Processors", *Proceedings of the 26th International Conference on Machine Learning*, pp. 873–880, 2009.

11. Bengio, Y., "Deep Learning of Representations: Looking Forward", *International Conference on Statistical Language and Speech Processing*, pp. 1–37, Springer, 2013.

12. Bengio, Y., N. Léonard and A. Courville, "Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation", *arXiv Preprint arXiv:1308.3432*, 2013.

13. Murdock, C., Z. Li, H. Zhou and T. Duerig, "Blockout: Dynamic Model Selection for Hierarchical Deep Networks", *2016 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2583–2591, IEEE, Las Vegas, NV, USA, 2016.

14. Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", *Journal of Machine Learning Research*, Vol. 15, No. 56, pp. 1929–1958, 2014.

15. Bengio, E., P.-L. Bacon, J. Pineau and D. Precup, "Conditional Computation in Neural Networks for Faster Models", *arXiv Preprint arXiv:1511.06297*, 2016.

16. Wu, Z., T. Nagarajan, A. Kumar, S. Rennie, L. S. Davis, K. Grauman and R. Feris, "BlockDrop: Dynamic Inference Paths in Residual Networks", *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8817–8826, IEEE, 2018.

17. Veit, A. and S. Belongie, "Convolutional Networks with Adaptive Inference Graphs", *International Journal of Computer Vision*, Vol. 128, No. 3, pp. 730–741, 2020.

18. Wang, X., F. Yu, Z.-Y. Dou, T. Darrell and J. E. Gonzalez, "Skipnet: Learning Dynamic Routing in Convolutional Networks", *Proceedings of the European Conference on Computer Vision*, pp. 409–424, 2018.

19. McGill, M. and P. Perona, "Deciding How to Decide: Dynamic Routing in Artificial Neural Networks", *International Conference on Machine Learning*, pp. 2363–2372, 2017.

20. Figurnov, M., M. D. Collins, Y. Zhu, L. Zhang, J. Huang, D. Vetrov and R. Salakhutdinov, "Spatially Adaptive Computation Time for Residual Networks", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1039–1048, 2017.

21. Liu, H., S. Parajuli, J. Hostetler, S. Chai and B. Bhanu, "Dynamically Throttleable Neural Networks (TNN)", *arXiv Preprint arXiv:2011.02836*, 2020.

22. Cai, S., Y. Shu and W. Wang, "Dynamic Routing Networks", *2021 IEEE Winter Conference on Applications of Computer Vision*, pp. 3587–3596, IEEE, 2021.

23. Jang, E., S. Gu and B. Poole, "Categorical Reparameterization with Gumbel-Softmax", *arXiv preprint arXiv:1611.01144*, 2016.

24. Maddison, C. J., A. Mnih and Y. W. Teh, "The Concrete Distribution: A Continuous Relaxation of Discrete Random variables", *arXiv preprint arXiv:1611.00712*, 2016.

25. Ioannou, Y., D. Robertson, D. Zikic, P. Kontschieder, J. Shotton, M. Brown and A. Criminisi, "Decision Forests, Convolutional Networks and the Models In-between", *arXiv Preprint arXiv:1603.01250*, 2016.

26. Teerapittayanon, S., B. McDanel and H.-T. Kung, "BranchyNet: Fast Inference via Early Exiting from Deep Neural Networks", *23rd International Conference on Pattern Recognition*, pp. 2464–2469, IEEE, 2016.

27. Jiang, Y.-G., C. Cheng, H. Lin and Y. Fu, "Learning Layer-Skippable Inference Network", *IEEE Transactions on Image Processing*, Vol. 29, pp. 8747–8759, 2020.

28. Eigen, D., M. Ranzato and I. Sutskever, "Learning Factored Representations in a Deep Mixture of Experts", *arXiv preprint arXiv:1312.4314*, 2013.

29. Shazeer, N., A. Mirhoseini, K. Maziarz, A. Davis, Q. V. Le, G. E. Hinton and J. Dean, "Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer", *Proceedings of the International Conference on Learning Representations*, International Conference on Learning Representations, 2017.

30. Ahmed, K., M. H. Baig and L. Torresani, "Network of Experts for Large-Scale Image Categorization", *European Conference on Computer Vision*, pp. 516–532, Springer, 2016.

31. Yan, Z., H. Zhang, R. Piramuthu, V. Jagadeesh, D. DeCoste, W. Di and Y. Yu, "HD-CNN: Hierarchical Deep Convolutional Neural Networks for Large Scale Visual Recognition", *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2740–2748, 2015.

32. Murthy, V. N., V. Singh, T. Chen, R. Manmatha and D. Comaniciu, "Deep Decision Network for Multi-class Image Classification", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2240–2248, 2016.

33. Kontschieder, P., M. Fiterau, A. Criminisi and S. R. Bulo, "Deep Neural Decision Forests", *2015 IEEE International Conference on Computer Vision*, pp. 1467–1475, IEEE, Santiago, Chile, 2015.

34. Zhou, Z.-H. and J. Feng, "Deep Forest: Towards an Alternative to Deep Neural

Networks", *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pp. 3553–3559, 2017.

35. LeCun, Y., B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard and L. D. Jackel, "Backpropagation Applied to Handwritten Zip Code Recognition", *Neural Computation*, Vol. 1, No. 4, pp. 541–551, 1989.

36. Ioffe, S. and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift", *International Conference on Machine Learning*, pp. 448–456, PMLR, 2015.

37. Ba, J. L., J. R. Kiros and G. E. Hinton, "Layer Normalization", *arXiv Preprint arXiv:1607.06450*, 2016.

38. Williams, R. J., "Simple Statistical Gradient-following Algorithms for Connectionist Reinforcement Learning", *Machine learning*, Vol. 8, No. 3, pp. 229–256, 1992.

39. Quinlan, J. R., "Induction of Decision Trees", *Machine learning*, Vol. 1, No. 1, pp. 81–106, 1986.

40. Quinlan, J. R., "C4.5: Programs for Machine Learning", *In Proceedings of 10th International Conference on Machine Learning*, pp. 252–259, 1993.

41. LeCun, Y., B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard and L. Jackel, "Handwritten Digit Recognition with a Back-Propagation Network", *Advances in Neural Information Processing Systems*, Vol. 2, 1989.

42. Lloyd, S., "Least Squares Quantization in PCM", *IEEE Transactions on Information Theory*, Vol. 28, No. 2, pp. 129–137, 1982.

43. Rand, W. M., "Objective Criteria for the Evaluation of Clustering Methods", *Journal of the American Statistical Association*, Vol. 66, No. 336, pp. 846–850, 1971.

44. Kingma, D. P. and J. Ba, "Adam: A Method for Stochastic Optimization", *arXiv Preprint arXiv:1412.6980*, 2014.