End-to-End Optimization and Learning for Multiagent Ensembles

James Kotary*
Syracuse University
Syracuse, United States of America
jkotary@syr.edu

Vincenzo Di Vito*
Syracuse University
Syracuse, United States of America
vdivitof@syr.edu

Ferdinando Fioretto
Syracuse University
Syracuse, United States of America
ffiorett@syr.edu

ABSTRACT

Multiagent ensemble learning is an important class of algorithms aimed at creating accurate and robust machine learning models by combining predictions from individual agents. A key challenge for the design of these models is to create effective rules to combine individual predictions for any particular input sample.

This paper addresses this challenge and proposes a unique integration of constrained optimization and learning to derive specialized consensus rules to compose accurate predictions from a pretrained ensemble. The resulting strategy, called *end-to-end Multiagent ensemble Learning e2e-MEL*, learns to select appropriate predictors to combine for a particular input sample. The paper shows how to derive the ensemble learning task into a differentiable selection program which is trained end-to-end within the ensemble learning model. Results over standard benchmarks demonstrate the ability of e2e-MEL to substantially outperform conventional consensus rules in a variety of settings.

KEYWORDS

Ensemble multi-agent learning, decision focused learning, integration of optimization and learning.

1 INTRODUCTION

Deep neural networks (DNNs) represent the current state-of-the art in a variety of machine learning tasks, and are finding utility in new applications at a rapid pace. Despite their effectiveness in pattern recognition tasks such as image classification and language modeling, they are subject to unavoidable error in deployment, due to phenomena such as overfitting and convergence to local minima in training. Even models with very high accuracy produce faulty classifications and erroneous predictions on some input samples, but similarly constructed models may fail on different input samples depending on differences of network architecture, training algorithm, or distribution of training data.

A variety of techniques have been presented in the machine learning literature to address this issue. Of particular relevance, *multiagent ensemble learning* [27] is a meta-algorithm which aims to create accurate and robust machine learning models by collecting and combining the outputs of individually pre-trained models. These individual *ensemble agents* or *members*, despite being trained to perform the same task, may exhibit error diversity, i.e., failure on separate samples so that their accuracy profiles complement each other across an overall distribution of test samples. The potential effectiveness of an ensemble model strongly depends on the correlation between the agents' errors across input samples and on their accuracy; those with higher accuracy and error diversity have higher potential for improved ensemble accuracy [21].

However, identifying the best model(s) among an ensemble for classifying any particular input sample is itself a nontrivial task. Most prior approaches rely on aggregating predictions across all agents of an ensemble, in order to make predictions that are more robust to the error of individual agents. These techniques could, in principle, be enhanced by applying them selectively to a subset of agents in the ensemble which is known to be more reliable on certain inputs. However, the design of algorithms that select and combine the agents' individual predictions in an effective way is a non-trivial endeavor. Consensus rule-based methods inevitably need to apply aggregation schemes that combine or exclude agents' predictions based on static rules, and thus miss an opportunity to inform the ensemble selection based a particular input's features.

Contributions. This paper focuses on addressing that difficulty and makes the following contributions: **(1)** It proposes an ensemble learning framework based on an integration of machine learning and optimization to derive a consensus mechanism to compose accurate predictions from a pretrained ensemble. The proposed strategy, called *end-to-end Multiagent Ensemble Learning (e2e-MEL)* learns to identify an appropriate *sub-ensemble* from which predictions are collected and combined *for a particular input sample*. **(2)** It shows how to cast the selection and aggregation of ensemble agent predictions as a differentiable optimization problem, which is parameterized by a deep neural network and trained end-to-end within the ensemble learning model. **(3)** An analysis on challenging learning tasks demonstrates the strengths of this idea: e2e-MEL is found to outperform models that attempt to select individual ensemble members as well as conventional consensus rules typically applied in ensemble learning, implying much higher ability to leverage error diversity than conventional consensus rules.

These results are significant: they demonstrate the integration of constrained optimization and learning to be a key enabler to construct accurate multiagent ensemble learners as a step forward to enhance error robustness in modern machine learning tasks.

To the best of the authors' knowledge, this paper presents the first adoption of symbolic learning to learn multiagent ensembles.

2 RELATED WORK

This paper proposes a framework for composing effective ensemble learning models by adapting techniques from the intersection of constrained optimization and machine learning. This section briefly reviews the two as-yet distinct areas:

2.1 Ensemble Learning

Several works have focused on studying ensemble models for more effective machine learning. We refer the reader to Dong et al. [9]

for a useful categorization of ensemble methods, depending on the underlying machine learning task to be solved.

Ensemble learning often involves two distinct aspects: **(1)** the training of individual ensemble learning agents, and **(2)** the aggregation of their individual outputs into accurate ensemble predictions. The former aspect concerns the composition of an ensemble from base agents with complementary error profiles, and is commonly handled by *bagging*, which consists of randomly composing datasets for training each ensemble member, and *boosting*, which involves adaptively contriving a sequence of datasets based on the error distributions of their resulting models so that error diversity is increased. Many variations and task-specific alternatives have also been proposed [9]. For example, Martínez-Muñoz et al. [20] studies the effects of various pruning techniques for reducing the ensemble size from an initial bagging ensemble. A survey focusing on the training of individual ensemble agents is provided in [21].

The latter aspect is typically handled by classical aggregation rules over either the discrete class predictions or the continuous activation values of ensemble members. *Majority* and *plurality voting* are consensus criteria for choosing a discrete class prediction based on the most popular among ensemble members. Some works have attempted mathematically model more effective aggregation rules: these include *super-learners* [15], which attempt to form a weighted combination of agent models that maximizes accuracy over a validation set.

This paper addresses the latter, challenging, aspect of ensemble modeling: optimizing the aggregation of predictions from individual ensemble agents. In contrast to other methods, the e2e-MEL approach aims to learn aggregation rules adaptively at the level of individual input samples, rather than a single rule for all samples. This unique feature allows for aggregated predictions to be informed by the samples' input features with enhanced performance, as the paper shows in Section 5.

2.2 Decision Focused Learning

A body of work has been devoted to studying constrained optimization problems as learnable components within neural networks. See for example the recent survey of Kotary et al. [17]. As a mapping between some problem-defining parameters and their optimal solutions, an optimization problem can be viewed as a function whose outputs adhere to explicitly defined constraints. In this way, special structure can be guaranteed within the predictions [10] or learned embeddings of a neural network [3]. The primary challenge is that differentiation of the optimization mapping is required for backpropagation. Distinct approaches have been proposed, which depend on various approximations, depending on the optimization problem's form [10], [23], and [11]. Smooth convex problems, in particular, admit exact gradients via their well-defined equations for optimality, as shown by Amos and Kolter [3] and Agrawal et al. [2]. Convex problems which define piecewise-constant mappings, such as linear programs, require approximation by smooth problems before differentiation. Smoothing techniques are commonly applied to the objective function and include the addition of regularizing objective terms, as proposed by Wilder et al. [26], and random noise, as in Berthet et al. [5].

In this paper, the desired structured prediction primarily takes the form of a binary vector representing subset selection, in which $k < n$ among n items are indicated for selection. This work exploits the fact that differentiation of this structure as a linear program with respect to a parameterizing vector allows it to be applied in an end-to-end differentiable masking scheme that is used to select and combine the best ensemble agents' predictions for a given input sample.

3 SETTING AND GOALS

In the present setting, an *ensemble* consists of a collection of n agents' models or *base learners* represented by functions $f_i, 1 \leq i \leq n$, trained independently on separate (but possibly overlapping) datasets $(\mathcal{X}_i,\mathcal{Y}_i)$, all on the same intended *classification* task. On every task studied, it assumed that $(\mathcal{X}_i,\mathcal{Y}_i)$ are given, along with a prescription for training each agents' models, so that f_i are assumed to be pre-configured. This setting is common in federated analytic contexts, where agents are often trained over a diverse body of datasets with skewed distributions [16], and in machine learning services, which provide pre-trained models chosen over an array of proprietary architectural and hyper-parametrization choices that often vary from service to service [24].

Let $n \in \mathbb{N}$ be the number of ensemble agents, $c \in \mathbb{N}$ the number of classes and $d \in \mathbb{N}$ the input feature size. Given a sample $z \in \mathbb{R}^d$, each base learner $f_j \colon \mathbb{R}^d \to \mathbb{R}^c$ computes $f_j(z) = \hat{y}_j$. For the classification tasks considered in this paper, each $\hat{y}_j$ is the direct output of a *softmax* function $\mathbb{R}^n \to \mathbb{R}^n$:

$$\texttt{softmax}(c)_i = \frac{e^{c_i}}{\sum_{k=1}^{n} e^{c_k}}. \tag{1}$$

Explicitly, each classifier $f_i(\phi_i, x)$ is trained with respect to its parameters ϕ_i to minimize a classification loss $\mathcal{L}$ as

$$\min_{\phi_i} \mathbb{E}_{(x,y)\sim(\mathcal{X}_i,\mathcal{Y}_i)} \left[\mathcal{L}(f_i(\phi_i, x), y)\right]. \tag{2}$$

The goal is then to combine the agents into a *smart ensemble*, whose aggregated classifier g performs the same task, but with greater overall accuracy on a *master dataset* $(\mathcal{X},\mathcal{Y})$, where $\mathcal{X}_i \subset \mathcal{X}$ and $\mathcal{Y}_i \subset \mathcal{Y}$ for all i with $0 \leq i \leq n$:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim(\mathcal{X},\mathcal{Y})} \left[\mathcal{L}(g(\theta, x), y)\right]. \tag{3}$$

As is typical in ensemble learning, the agents may be trained in a way that increases test-error diversity among f_i on $\mathcal{X}$ — see Section 5. In each dataset there is implied a train/test/validation split, so that evaluation of a trained model is always performed on its test portion. Where this distinction is needed, the symbols $\mathcal{X}_{\texttt{train}}, \mathcal{X}_{\texttt{valid}}, \mathcal{X}_{\texttt{test}}$ are used.

Table 1 contains a list of symbols used to describe various aspects of the computation, along with their meanings.

4 END-TO-END MULTIAGENT ENSEMBLE LEARNING (E2E-MEL)

Ideally, given a pretrained ensemble $f_i, 1 \leq i \leq n$ and a sample $z \in \mathcal{X}$, one would select from the ensemble a classifier which is known to produce an accurate class prediction for z. However, a performance assessment for each agents' predictions is not available

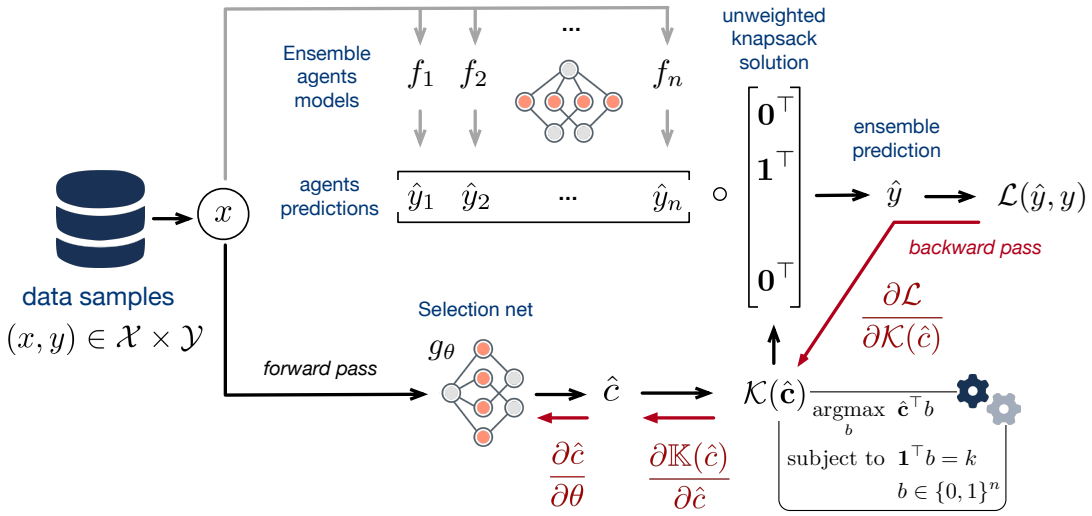

Figure 1: End-to-end Smart Multi Agent Ensemble learning scheme: Black and red arrows illustrate forward and backward operations, respectively.

Symbol	Semantic
n	Size of the ensemble
k	Size of a sub-ensemble
m	Number of noise samples for perturbed optimizer
f_i	Pre-trained classifier, $1 \leq i \leq n$
$\hat{y}_i$	Softmax prediction from f_i
f	Smart ensemble classifier
$\hat{y}$	Smart ensemble softmax prediction
g	Selection net
θ	Selection net parameters
$\hat{c}$	Predicted scores from the selection net
$\mathcal{K}$	Unweighted knapsack function
$\mathbb{K}$	Perturbed unweighted knapsack function
$\mathcal{X}$	Input feature data
$\mathcal{Y}$	Target class data
$\mathcal{L}$	Classification loss function

Table 1: Common symbols

at test time. Thus, conventional ensemble learning schemes resort to selection criteria such as majority voting, which is defined as follows: given agents' class predictions $h_i, 1 \leq i \leq n$ represented as binary one-hot vectors, the majority vote is equal to the most-predicted class; equivalently, $\text{argmax}\left(\sum_{i=1}^{n} h_i\right)$.

The end-to-end learning scheme in this work is motivated by the intuition that a more useful ensemble prediction could instead be informed by prediction based on z, and that voting over a well-selected sub-ensemble can provide more accurate inferences than voting over the entire master ensemble. Further, it can do so more reliably than a well-selected single base learner can. The sub-ensemble size k is treated as a hyperparameter. While a natural choice would be to select only the best predicted agent (thus, setting $k = 1$) for a given input sample, it is consistently observed (as the paper shows in Section 5) that optimal task performance is achieved for a value of k that is strictly larger than 1 but smaller than n.

The proposed mechanism casts the sub-ensemble selection as an optimization program that is end-to-end differentiable and can thus be integrated with a learning model g_θ to select a reliable subset of the ensemble agents to combine for predictions. An *end-to-end Smart Multi Agent Ensemble* (e2e-MEL), or simply, *smart ensemble*, consists of an ensemble of agent models along with a module which is trained by e2e-MEL to select the *sub-ensemble*, of specified size k, which produces the most accurate combined prediction for a given input. The model g is called the *selection net*, and the end-to-end ensemble model is trained by optimizing its parameters θ.

The e2e-MEL model is composed of three main steps:

(1) Predict a vector of scores $g_\theta(z) = \hat{c}$, estimating a notion of prediction accuracy for each base learner on sample z.

(2) Identify the base learner indices $\mathcal{E} \subset [n]$ which correspond the the top k predicted scores.

(3) Collect the predictions of the selected sub-ensemble: $f_j(z), j \in \mathcal{E}$ and perform an approximate majority voting scheme over those predictions to determine the class of z.

By training on the master set $\mathcal{X}_{\texttt{train}}$, the smart ensemble learns to make better predictions by virtue of learning to select better sub-ensembles to vote on its input samples. However, note that subset selection and majority voting are discrete operations, and in plain form do not offer useful gradients for backpropagation and learning. The next sections discuss further details of the e2e-MEL framework, including differentiable approximations for each step of the overall model.

Figure 1 illustrates the e2e-MEL model and training, in terms of its component operations. Backpropagation is illustrated in backward red arrows; note that only operations downstream from the selection net g are backpropagated, since the e2e-MEL is parameterized by the parameters of g alone.

4.1 Differentiable Model Selection

The e2e-MEL system is based on learning to select $k < n$ predictions from the master ensemble, given a set of input features. This can be done by way of a structured prediction of binary values, which are then used to mask the individual agent predictions.

Consider the unweighted knapsack problem

$$\mathcal{K}(\hat{c}) = \underset{b}{\text{argmax}} \quad \hat{c}^\top b \tag{4a}$$

$$\text{subject to} \quad \mathbf{1}^\top b = k, \tag{4b}$$

$$b \in \{0,1\}^n, \tag{4c}$$

which can be viewed as a selection problem whose optimal solution assigns the value 1 to the elements of b associated to the top k values of $\hat{c}$. Relaxing the constraint (4c) to $0 \leq b \leq 1$ results in an equivalent linear programming problem with discrete optimal solutions $b \in \{0,1\}^n$, despite being both convex and composed of continuous functions. This useful property holds for any linear program with totally unimodular constraints and integer right-side-side coefficients [4].

This optimization problem can be viewed as a mapping from $\hat{c}$ to a binary vector indicating its top k values, and represents thus a natural candidate for selection of the optimal sub-ensemble of size k given the individual agents' predicted scores, seen as $\hat{c}$. The outputs of Problem (4) define a piecewise constant function, $\mathcal{K}(\hat{c})$, which does not admit readily informative gradients, posing challenges to differentiability. For integration into the end-to-end learning system, the function $\mathcal{K}(\hat{c})$ must provide informative gradients with respect to $\hat{c}$. In this work, this challenge is overcome by smoothing $\mathcal{K}(\hat{c})$ based on perturbing $\hat{c}$ with random noise, as elaborated next.

As observed in Berthet et al. [5], any continuous, convex linear programming problem can be used to define a *differentiable perturbed optimizer*, which yields approximately the same solutions but is differentiable with respect to its linear objective coefficients. Given a random noise variable Z with probability density $p(z) \propto \exp(-v(z))$ where v is a twice differentiable function,

$$\mathbb{K}(\hat{c}) = \mathbb{E}_{z\sim Z} \quad [\mathcal{K}(\hat{c} + \epsilon z)], \tag{5}$$

is a differentiable perturbed optimizer associated to $\mathcal{K}$. The factor $\epsilon > 0$ is a temperature parameter which controls the sensitivity of its gradients (or properly, Jacobian matrix), which can itself be represented by the expected value [1]:

$$\frac{\partial \mathbb{K}(\hat{c})}{\partial \hat{c}} = \mathbb{E}_{z\sim Z} \quad \left[\mathcal{K}(\hat{c} + \epsilon z) \;\; v'(z)^\top\right]. \tag{6}$$

In this work, Z is modeled as a standard normal random variable. While these expected values are analytically intractable (due to the constrained argmax operator within $\mathcal{K}$), they can be estimated to arbitrary precision by sampling in Monte Carlo fashion. This procedure is a generalization of the Gumbel Max Trick [13].

Note that simulating Equations (5) and (6) requires solving Problem (4) for potentially many values of z. However, although the theory of perturbed optimizers requires the underlying problem to be a linear program, only a blackbox implementation is required to produce $\mathcal{K}(\hat{c})$ [5], allowing for an efficient algorithm to be used in place of a (more costly) linear programming solver. The complexity of evaluating the differentiable perturbed optimizer $\mathbb{K}(\hat{c})$ is discussed next.

Theorem 1. *For the ensemble size n, and sub-ensembles of size k, the total computation required for solving Problem* (4) *is $O(n \cdot \log k)$.*

Proof. This result relies on the observation that $\mathcal{K}(\hat{c})$ can be computed efficiently by identifying the top k values of $\hat{c}$ in $O(n \log k)$ time using a divide-and-conquer technique. See, for example, Cormen et al. [6]. □

Generating m such solutions for gradient estimation then requires total computation $O(m \cdot n \log k)$. Note, however, that these operations can be performed in parallel across samples, allowing for sufficient noise samples to be easily generated for computing accurate gradients, especially when GPU computing is available.

For clarity, note also that the function $\mathcal{K}$, as a linear program mapping, has a discrete output space since any linear program takes its optimal solution at a vertex of its feasible region [4], which are finite in number. As such, it is a piecewise constant function and is differentiable except on a set of measure zero [12]. However, $\frac{\partial \mathcal{K}}{\partial \hat{c}} = 0$ everywhere it is defined, so the derivatives lack useful information for gradient descent [26]. While $\frac{\partial \mathbb{K}}{\partial \hat{c}}$ is not the true derivative of $\mathcal{K}$ at $\hat{c}$, it supplies useful information about its direction of descent.

In practice, the forward optimization pass is modeled as $\mathcal{K}(\hat{c})$, and the backward pass is modeled as $\frac{\partial \mathbb{K}(\hat{c})}{\partial \hat{c}}$. This allows further downstream operations, and their derivatives, to be evaluated at $\mathcal{K}(\hat{c})$ without approximation, which improves training and test performance.

These forward and backward passes together are henceforth referred to as the *Knapsack Layer*. Its explicit backward pass is computed as

$$\frac{\partial \mathbb{K}(\hat{c})}{\partial \hat{c}} \approx \frac{1}{m} \sum_{i=1}^{m} \quad \left[\mathcal{K}(\hat{c} + \epsilon z_i) \;\; v'(z_i)^\top\right], \tag{7}$$

where $z_i \sim \mathcal{N}(0,1)^n$ are m independent samples each drawn from a standard normal distribution.

4.2 Combining Predictions

Denote as $P \in \mathbb{R}^{c\times n}$ the matrix whose $j-th$ column is the softmax vector $\hat{y}_j$ of base learner j,

$$P = \begin{pmatrix} \hat{y}_1 & \hat{y}_2 \ldots \hat{y}_n. \end{pmatrix} \tag{8}$$

For the purpose of combining the ensemble agent predictions, $\mathcal{K}(\hat{c})$ is treated as a binary masking vector $b \in \{0,1\}^n$, which selects the subset of agents for making a prediction. Denote as $B \in \{0,1\}^{c\times n}$ the matrix whose i^{th} column is $B_i = \overrightarrow{\mathbf{1}}\, b_i$; i.e.,

$$B = \begin{bmatrix} b^\top \\ \vdots \\ b^\top \end{bmatrix}.$$

This matrix is used to mask the agent models' softmax predictions P by element-wise multiplication. Next, define

$$\begin{aligned} P_k &= B \circ P \\ &= \begin{bmatrix} b^\top \\ \vdots \\ b^\top \end{bmatrix} \circ \begin{bmatrix} \hat{y}_1 & \ldots & \hat{y}_n \end{bmatrix} \end{aligned} \tag{9}$$

Doing so allows to compute the sum of predictions over the selected sub-ensemble $\mathcal{E}$, but in a way that is automatically differentiable, that is:

$$\hat{v} \coloneqq \sum_{i \in \mathcal{E}} \hat{y}_i = \sum_{i=1}^{n} P_k^{(i)}. \tag{10}$$

The smart ensemble prediction comes from applying softmax to this sum:

$$\begin{aligned} \hat{y} &= \texttt{softmax}(\hat{v}) \\ &= \texttt{softmax}\left(\sum_{i=1}^{n} P_k^{(i)}\right), \end{aligned} \tag{11}$$

viewing the softmax as a smooth approximation to the argmax function as represented with one-hot binary vectors. This function is interpreted as a smoothed majority voting to determine a class prediction: given one-hot binary class indicators h_i, the majority vote is equal to $\text{argmax}(\sum_i h_i)$.

At test time, class predictions are calculated as

$$\underset{1 \le i \le c}{\text{argmax}} \;\; \hat{y}_i(x). \tag{12}$$

Combining predictions in this way allows for an approximated majority voting over a selected sub-ensemble, but in a differentiable way so that selection net parameters θ can be directly trained to produce selections that minimize the classification task loss, as detailed in the next section.

4.3 Learning Selections

The smart ensemble mechanism learns more accurate class predictions by learning to select better subensembles to vote on its input samples. In turn, this is done by predicting better coefficients $\hat{c}$ which parameterize the Knapsack Layer.

The task of predicting $\hat{c}$ based on input z is itself learned by the *selection net*, a neural network model g so that $\hat{c} = g(z)$. Since g acts on the same input samples as each f_i, it should be capable of processing inputs from z at least as well as the agents' models; in Section 5, the selection net in each experiment uses the same CNN architecture as that of the agent models. Its predicted values are viewed as scores over the ensemble members, rather than over the possible classes. High scores correspond to agents which are well-qualified to vote on the sample in question.

In practice, the selection net's predictions $\hat{c}$ are normalized before input to the mapping $\mathcal{K}$:

$$\hat{c} \leftarrow \frac{\hat{c}}{\|\hat{c}\|_2}. \tag{13}$$

This has the effect of standardizing the magnitude of the linear objective term in (4a), and tends to improve training. Since scaling the objective of an optimization problem has no effect on the solution, this is equivalent to standardizing the relative magnitudes of the linear objective and random noise perturbations in Equations (5) and (6), preventing ϵ from being effectively absorbed into the predicted $\hat{c}$.

For training input x, let $\hat{y}_\theta(x)$ represent the associated e2e-MEL prediction given the selection net parameters θ. During training, the classification loss between these predictions and ground-truth labels is minimized:

$$\min_\theta \mathbb{E}_{(x,y)\sim(\mathcal{X},\mathcal{Y})} \left[\mathcal{L}(\hat{y}_\theta(x), y)\right]. \tag{14}$$

Generally, the loss $\mathcal{L}$ is chosen to be the same as that for training the agent models, since the classification task is the same as that which the individual agents have been trained to perform.

4.4 e2e-MEL Algorithm Details

Algorithm 1 summarizes the e2e-MEL procedure for training a selection net. Note that only the parameters of the selection net are optimized in training, and so only its downstream computations are backpropagated. This is done by the standard automatic differentiation employed in machine learning libraries [22], except in the case of the Knapsack Layer, whose gradient transformation is analytically specified by Equation (7).

For clarity, Algorithm 1 is written in terms of operations that apply to a single input sample. In practice however, minibatch gradient descent is used, as is typical in machine learning [14]. Each pass of the training begins evaluation of each agent model (lines 3 and 4), and the sampling of standard normal random noise vectors (lines 5 and 6). The selection net predicts from input features x a vector of agent scores $\hat{c}$ (line 7), which define an unweighted knapsack problem $\mathcal{K}(\hat{c})$ that is solved to produce the binary mask b (line 8).

Masking is applied to the agent predictions before being summed and softmaxed for a final ensemble prediction $\hat{y}$ (line 10–12). The classification loss $\mathcal{L}$ is evaluated with respect to the label y (line 13) and backpropagated in three steps: **(1)** The gradient $\frac{\partial L}{\partial b}$ is computed by automatic differentiation backpropagated to the Knapsack Layer's output (line 14). **(2)** The chain rule factor $\frac{\partial b}{\partial c}$ is analytically computed by the methodology of Section 4.1 (line 15). **(3)** The remaining chain rule factor $\frac{\partial \hat{c}}{\partial \theta}$ is computed by automatic differentiation (line 16). Note that as each chain rule factor is computed, it is also applied toward computing $\frac{\partial L}{\partial \theta}$ (line 17). Finally, a stochastic gradient descent step [25] or alternatively one of its variants ([8], [28]) is applied to update θ (line 18).

The next section evaluates the accuracy of ensemble models trained with this algorithm, on classification tasks using deep neural networks.

5 E2E-MEL EVALUATION

The e2e-MEL training is evaluated on several vision classification tasks: digit classification on *MNIST dataset* [7], age-range estimation on *UTKFace dataset* [29], image classification on *CIFAR10 dataset* [18], and emotion detection on *FER2013* dataset [19]. The experiments are primarily focused on the efficacy of e2e-MEL training as compared with the following widely adopted baseline aggregation rules when paired with a pre-trained ensemble:

- *Unweighted Average*, which averages all the agents' softmax predictions and then computing the index of the corresponding highest label score as the final prediction;
- *Majority Voting*, which makes a discrete class prediction from each agent and then returns the most-predicted class;
- *Random Selection*, which randomly selects a size-k sub-ensemble of agents for making prediction and then applies

Algorithm 1: Training the Selection Net

input : $\mathcal{X}, \mathcal{Y}\, \alpha, k, m, epsilon$

for *epoch* $k = 0, 1, \ldots$ **do**
  foreach $(\boldsymbol{x}, \boldsymbol{y}) \leftarrow (\mathcal{X}, \mathcal{Y})$ **do**
    foreach $1 \le i \le n$ **do**
      $\hat{y}_i \leftarrow f_i(\boldsymbol{x})$
    foreach $1 \le j \le m$ **do**
      $z_j \sim \mathcal{N}(0,1)^n$
    $\hat{c} \leftarrow g_{\boldsymbol{\theta}}(\boldsymbol{x})$
    $b \leftarrow \mathcal{K}(\hat{c})$
    $\hat{c} \leftarrow \frac{\hat{c}}{\|\hat{c}\|_2}$
    $P_k \leftarrow \begin{bmatrix} b & \ldots & b \end{bmatrix}^\top \circ \begin{bmatrix} \hat{y}_1 & \ldots & \hat{y}_n \end{bmatrix}$
    $\hat{v} \leftarrow \sum_{i=1}^{n} P_k^{(i)}$
    $\hat{p} \leftarrow \texttt{softmax}(\hat{v})$
    $L \leftarrow \mathcal{L}(\hat{p}, y)$
    $\frac{\partial L}{\partial b} \leftarrow \texttt{autodiff}$
    $\frac{\partial b}{\partial \hat{c}} \leftarrow \frac{1}{m} \sum_{i=1}^{m} \begin{bmatrix} \mathcal{K}(\hat{c} + \epsilon z_i) & v'(z_i)^\top \end{bmatrix}$
    $\frac{\partial \hat{c}}{\partial \theta} \leftarrow \texttt{autodiff}$
    $\frac{\partial L}{\partial \theta} \leftarrow \frac{\partial L}{\partial b} \cdot \frac{\partial b}{\partial \hat{c}} \cdot \frac{\partial \hat{c}}{\partial \theta}$
    $\theta \leftarrow \theta - \alpha \frac{\partial L}{\partial \theta}$

the unweighted average rule to the selected agents' soft predictions.

As described in Section 1, ensemble learning schemes are most effective when agent models are accurate and have high error diversity. In this section, agents are deliberately trained to have high error diversity with respect to input samples belonging to different classes. This is done by composing for each agent model f_i $(1 \le i \le n)$ a training dataset $\mathcal{X}_i$ in which a subset of classes is over-represented, resulting agents that specialize in identifying one or more classes. The exact class composition of each dataset $\mathcal{X}_i$ depends on the particular classification task and on the agent's intended specialization.

For each task, each agents is designed to be specialized for recognizing either one or two particular classes. To this end, the training set of each agent is partitioned to have a majority of samples belonging to a particular class, while the remaining part of the training dataset is uniformly distributed across all other classes by random sampling. Specifically, to compose the ensemble for each task, a single agent is trained to specialize on each possible class, and on each pair of classes (e.g., digits 1 and 2 in MNIST). When c is the number of classes, the experimental smart ensemble then consists of $c + \binom{c}{2}$ total agents. Training a specialized agent in this way generally leads to high accuracy over its specialty classes, but low accuracy over all other classes. Therefore in this experimental setup, no single agent is capable of achieving high overall accuracy on the master test set $\mathcal{X}_{\texttt{test}}$.

Table 2 shows the average accuracy of individual agent models on their specialty classes and their non-specialty classes; reported, respectively as *specialized accuracy* and *complementary accuracy*. The reported *overall* accuracy is measured over the entire master test set $\mathcal{X}_{\texttt{test}}$. This sets the stage for demonstrating the ability of e2e-MEL training to compose a classifier that substantially outperforms its base agent models on $\mathcal{X}_{\texttt{test}}$ by adaptively selecting sub-ensembles based on input features; see Section 5.2.

Dataset	Accuracy (%) Specialized	Complimentary	Overall
MNIST	97.5	27.8	40.6
UTKFACE	93.2	25.2	38.6
FER2013	79.4	37.1	45.4
CIFAR10	76.3	24.8	35.1

Table 2: Specialized agent model test accuracy

Note that, in each experiment, the agent models' architecture design, hyperparameter selection, and training methods have not been chosen to fully optimize classification accuracy, which is not the direct goal of this work. Instead the agents are trained to maximize error diversity, and demonstrate the ability of e2e-MEL to leverage error diversity and compose highly accurate ensemble models from far less accurate agent models, in a way that is not shared by conventional aggregation rules. Note also that improving agent model accuracies would, of course, tend to improve the accuracy of the resulting ensemble classifiers. In each case, throughout this section, the e2e-MEL selection net is given the same CNN architecture as the base agent models which form its ensemble.

5.1 Datasets and Settings

5.1.1 Digit Classification. MNIST is a 28x28 pixel greyscale images of handwritten digits dataset, which contains 60000 images for training and 10000 images for testing. Here the agents use convolutional neural networks (CNN), each of those with the same architecture, 2 convolutional layer and 3 linear layer. Within the convolutional layers, of 10 and 20 output channels respectively, the information is processed using a square kernel filter of size 5x5. Here the training dataset of each agent is composed, on average, by 73.2% of samples belonging to one (or two) specific class(es), while the remaining part is uniformly distributed over all the other classes by random sampling. On average, this results in 97.5% of accuracy in classifying a particular subset of training samples and 20% of accuracy in classifying the complementary subset of training samples. The ensemble consists of 55 agents, 10 of which specialize on 1 class and $\binom{10}{2} = 45$ of which specialize on 2 classes.

5.1.2 Image classification. CIFAR10 is a 32x32 pixel color images dataset in 10 classes: airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. It contains 6000 images of each class. Here the agents use CNN models, each of those with the same architecture, 2 convolutional layer, each containing a square kernel filter of size 5, with 6 and 16 output channels respectively, and 3 linear layer. On average, the training dataset of each agent is composed by 55.2% of samples belonging to one (or two) specific class(es), while the remaining part is uniformly distributed over all the other classes by random sampling. This results in 76.3% accuracy for classifying a particular subset of training samples and 24.8% accuracy for classifying the complementary subset of training

	Accuracy (%)			
Dataset	**e2e-MEL**	**UA**	**MV**	**RS**
MNIST	**98.55**	96.91	95.99	96.93
UTKFACE	**90.97**	84.60	80.78	84.60
FER2013	**66.31**	63.89	63.15	63.89
CIFAR10	**64.09**	60.59	60.35	60.59

Table 3: e2e-MEL vs unweighted average (UA), majority voting (MV), and random selection (RS), using specialized agents.

samples. The ensemble consists of 55 agents, 10 of which specialize on 1 class and $\binom{10}{2} = 45$ of which specialize on 2 classes.

5.1.3 Age estimation. UTKFace is a face images dataset which consists of over 20000 samples and different version of images format. Here, about 9700 cropped and aligned images are split in 5 classes: baby (up to 5 years old), teenager (up to 19), young (up to 33), adult (up to 56) and senior (more than 56 years old). The classes are not uniformly distributed per number of ages, but each class contains the same number of samples. The goal is to estimate the person age given the face image. On average, the training dataset of each agent is composed by 58.3% of samples belonging to one (or two) specific class(es), while the remaining part is uniformly distributed over all the other classes by random sampling. This results in 95.2% accuracy for classifying a particular subset of training samples and 25.2% accuracy for classifying the complementary subset of training samples. Here the agent model architecture is a pretrained version of Resnet18 on the ImageNet-1k dataset, entirely re-trained. The ensemble consists of 15 agents, 5 of which specialize on 1 class and $\binom{5}{2} = 10$ on 2 classes.

5.1.4 Emotion detection. Fer2013 is dataset containg over 30000 48x48 pixel grayscale images of faces of 7 classes: angry, disgust, fear, happy, neutral, sad and surprises. The goal is to categorize the emotion shown in the facial expression into one category. The training dataset of each agent is composed by 44.4% of samples belonging to one (or two) specific class(es), while the remaining part is uniformly distributed over all the other classes by random sampling. This results in 79.4% accuracy for classifying a particular subset of training samples and 37.1% accuracy for classifying the complementary subset of training samples. Here the agent model architecture consists of 8 convolutional layer and a linear layer. For each convolutional layer, the number of output channels is 64,128,128,128,256,512,512 and 512; each uses a square kernel of size 3x3. The ensemble consists of 21 agents, 7 of which specialize on 1 class and $\binom{7}{2} = 21$ of which specialize on 2 classes.

5.2 e2e-MEL Analysis

The *e2e-MEL* strategy is tested on each experimental task for sub-ensemble size k varying between 1 and the full ensemble size n, and compared to the baseline methods described above. Note in each case that accuracy is defined as the percentage of correctly classified samples over the master test set.

Table 3 reports the best accuracy over all the ensemble sizes k of ensembles trained by e2e-MEL along with that of each baseline ensemble model, where each are formed using the same pre-trained agents. In each task the e2e-MEL scheme outperforms all the baseline methods, for all but the lowest values of k.

Figure 2 reports the test accuracy found by the proposed e2e-MEL framework and ensembles based on a weighted average, majority voting, or random selection scheme. We make two key observations: **(1)** Note from each subplot in Figure 2 that smart ensembles of size $k > 1$ provide more accurate predictions than baseline models that randomly select sub-ensembles of the same size, a trend that diminishes as k increases and agent selections have less consequence (the two perform equally when $k = n$). **(2)** In every case, the sub-ensemble size which results in optimal performance is strictly between 1 and n. *Importantly, this illustrates the motivating intuition of the e2e-MEL ensemble training. Neither the full ensemble ($k = n$), nor smart selection of a single agent model ($k = 1$) can outperform models that use smart selection of a sub-ensemble of any size.* A well-selected sub-ensemble has higher potential accuracy than the master ensemble, and is, on average, more reliable than a well-selected single agent.

			Accuracy (%)	
Dataset	**Classes**	**Best k**	**e2e-MEL**	**Individual agents**
MNIST	10	10	**98.55**	40.6
UTKFACE	5	7	**90.97**	38.6
FER2013	7	13	**66.31**	45.4
CIFAR10	10	10	**64.09**	35.1

Table 4: Left: Best ensemble size (Best k) and associated e2e-MEL test accuracy attained on each dataset. Right: Average accuracy for the constituent ensemble agents.

Next, Table 4 (left) reports the accuracy of the e2e-MEL model trained on each task, along with the sub-ensemble size that resulted in highest accuracy. In two cases, for the digit classification task and the image classification task, the e2e-MEL performs best when the sub-ensemble size is equal to the number of classes. In the remaining tasks, this observation holds approximately. *This is intuitive, since the number of agents specializing on any class is equal to the number of classes, and e2e-MEL is able to increase ensemble accuracy by learning to select these agents for prediction.*

Finally, observe the accuracy of e2e-MEL in Table 4 (left) and the performance of the individual agents predictors of the ensemble tested on both the labels in which their training was specialized as well as the other labels. Note how e2e-MEL predictions outperform their constituent agents by a wide margin on each task. For example, on MNIST, the e2e-MEL ensemble reaches an accuracy 58 percentage points higher than its average constituent agent. *This illustrates the ability of e2e-MEL to leverage the error diversity of agents to form accurate classifiers by composing them based on input features*, even when the individual agent's accuracies are poor.

6 CONCLUSION

This paper was motivated by the desire to overcome one of the key challenges in multiagent ensemble learning: the design of effective rules to combine individual predictions for any particular input sample. To address this challenge, the paper proposed *end-to-end*

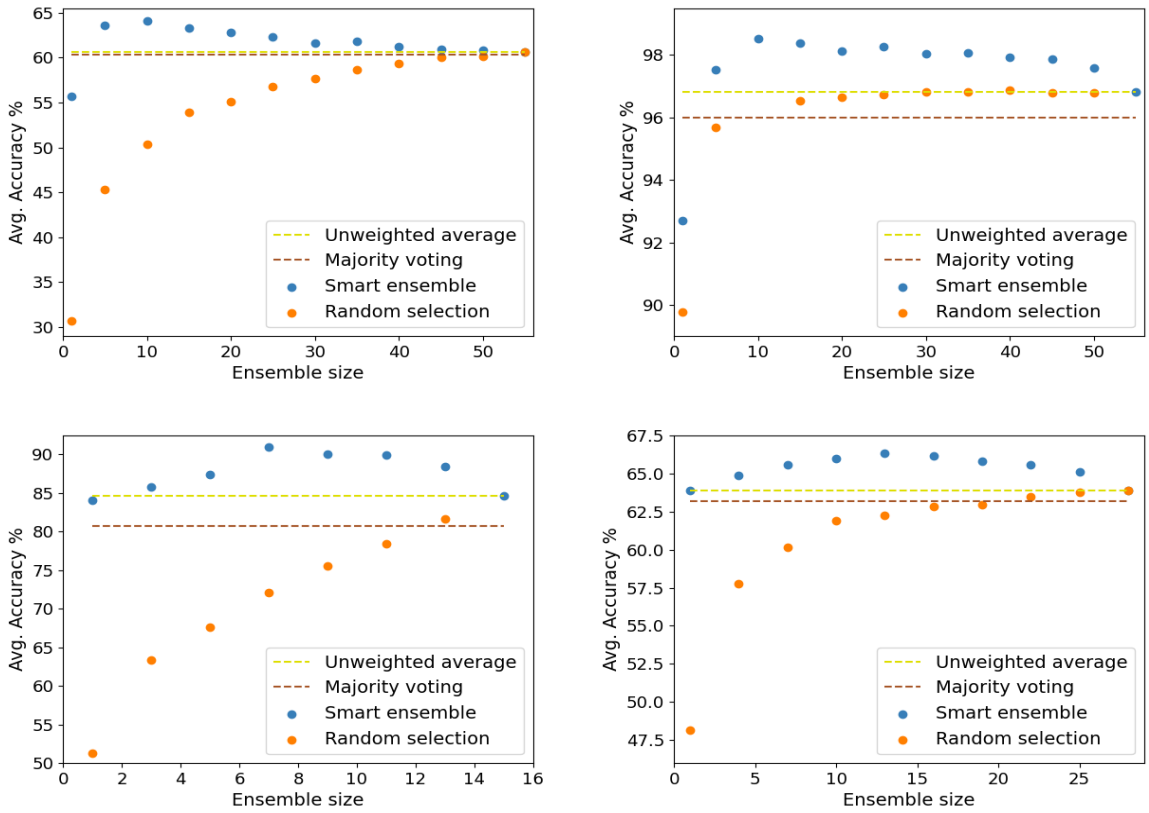

Figure 2: Comparison between e2e-MEL and other ensemble models at varying of the sub-ensemble size k on image classification–CIFAR10–(top left), digit classification–MNIST–(top right), age estimation–UTKFace–(bottom left), and emotion detection–FER2013–(bottom right).

Multiagent ensemble Learning e2e-MEL, a unique integration of constrained optimization and learning to derive specialized consensus rules to compose accurate predictions from a pretrained ensemble. Motivated by the intuition that well-selected sub-ensembles can form more accurate predictions than their master ensemble, it aims to adaptively select sub-ensembles according to individual input samples. The paper shows how to derive the ensemble learning task into a differentiable selection program which is trained end-to-end within the ensemble learning model. This strategy endows e2e-MEL with the ability to compose accurate classification models even from ensemble agents with low accuracy, a feature not shared by existing ensemble learning approaches. This demonstrates a noteworthy ability of e2e-MEL to leverage error diversity across ensemble models. Results over standard benchmarks demonstrate the ability of e2e-MEL to substantially outperform conventional consensus rules in a variety of settings.

This work demonstrates the integration of constrained optimization and machine learning models as a valuable toolset for not only enhancing but also combining machine learning models to improve performance on common tasks. We believe this is a very promising area and hope our work can motivate new solutions in which decision focused learning may be used to improve the capabilities of machine learning systems.

ACKNOWLEDGMENTS

This research is partially supported by NSF grant 2007164 and NSF CAREER Award 2143706. Its views and conclusions are those of the authors only.

REFERENCES

[1] J. Abernethy, C. Lee, and A. Tewari. Perturbation techniques in online learning and optimization. *Perturbations, Optimization, and Statistics*, 233, 2016.
[2] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and J. Z. Kolter. Differentiable convex optimization layers. *Advances in neural information processing systems*, 32, 2019.
[3] B. Amos and J. Z. Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, pages 136–145. PMLR, 2017.
[4] M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali. *Linear programming and network flows*. John Wiley & Sons, 2008.
[5] Q. Berthet, M. Blondel, O. Teboul, M. Cuturi, J.-P. Vert, and F. Bach. Learning with differentiable pertubed optimizers. *Advances in neural information processing systems*, 33:9508–9519, 2020.
[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT press, 2022.

[7] L. Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012. doi: 10.1109/MSP.2012.2211477.

[8] P. Diederik and B. Jimmy. Adam: A method for stochastic optimization. iclr. *arXiv preprint arXiv:1412.6980*, 2014.

[9] X. Dong, Z. Yu, W. Cao, Y. Shi, and Q. Ma. A survey on ensemble learning. *Frontiers Comput. Sci.*, 14(2):241–258, 2020. doi: 10.1007/s11704-019-8208-z. URL https://doi.org/10.1007/s11704-019-8208-z.

[10] A. N. Elmachtoub and P. Grigas. Smart "predict, then optimize". *Management Science*, 2021.

[11] A. Ferber, B. Wilder, B. Dilkina, and M. Tambe. Mipaal: Mixed integer program as a layer. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

[12] G. B. Folland. *Real analysis: modern techniques and their applications*, volume 40. John Wiley & Sons, 1999.

[13] E. J. Gumbel. *Statistical theory of extreme values and some practical applications: a series of lectures*, volume 33. US Government Printing Office, 1954.

[14] J. Heaton. Ian goodfellow, yoshua bengio, and aaron courville: Deep learning, 2018.

[15] C. Ju, A. Bibaut, and M. van der Laan. The relative performance of ensemble methods with deep convolutional neural networks for image classification. *Journal of Applied Statistics*, 45(15):2800–2818, 2018.

[16] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R. G. L. D'Oliveira, H. Eichner, S. E. Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konecný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, H. Qi, D. Ramage, R. Raskar, M. Raykova, D. Song, W. Song, S. U. Stich, Z. Sun, A. T. Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu, and S. Zhao. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021. ISSN 1935-8237. doi: 10.1561/2200000083. URL http://dx.doi.org/10.1561/2200000083.

[17] J. Kotary, F. Fioretto, P. Van Hentenryck, and B. Wilder. End-to-end constrained optimization learning: A survey. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 4475–4482, 2021. doi: 10.24963/ijcai.2021/610. URL https://doi.org/10.24963/ijcai.2021/610.

[18] A. Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[19] K. Liu, M. Zhang, and Z. Pan. Facial expression recognition with cnn ensemble. In *2016 International Conference on Cyberworlds (CW)*, pages 163–166, 2016. doi: 10.1109/CW.2016.34.

[20] G. Martínez-Muñoz, D. Hernández-Lobato, and A. Suárez. An analysis of ensemble pruning techniques based on ordered aggregation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):245–259, 2009. doi: 10.1109/TPAMI.2008.78.

[21] I. D. Mienye and Y. Sun. A survey of ensemble learning: Concepts, algorithms, applications, and prospects. *IEEE Access*, 10:99129–99149, 2022.

[22] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[23] M. V. Pogančić, A. Paulus, V. Musil, G. Martius, and M. Rolinek. Differentiation of blackbox combinatorial solvers. In *International Conference on Learning Representations (ICLR)*, 2020.

[24] M. Ribeiro, K. Grolinger, and M. A. Capretz. Mlaas: Machine learning as a service. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 896–902, 2015. doi: 10.1109/ICMLA.2015.152.

[25] S. Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[26] B. Wilder, B. Dilkina, and M. Tambe. Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization. In *AAAI*, volume 33, pages 1658–1665, 2019.

[27] I. H. Witten, E. Frank, M. A. Hall, C. J. Pal, and M. DATA. Practical machine learning tools and techniques. In *Data Mining*, volume 2, 2005.

[28] M. D. Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[29] S. Y. Zhang, Zhifei and H. Qi. Age progression/regression by conditional adversarial autoencoder. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017.