T^2-Net: A Semi-supervised Deep Model for Turbulence Forecasting

Denghui Zhang[1], Yanchi Liu[2*], Wei Cheng[2], Bo Zong[2], Jingchao Ni[2],
Zhengzhang Chen[2], Haifeng Chen[2], Hui Xiong[1]
[1]Rutgers University, USA, {denghui.zhang, hxiong}@rutgers.edu
[2]NEC Labs America, USA, {yanchi, weicheng, bzong, jni, zchen, haifeng}@nec-labs.com

***Abstract*—Accurate air turbulence forecasting can help airlines avoid hazardous turbulence, guide the routes that keep passengers safe, maximize efficiency, and reduce costs. Traditional turbulence forecasting approaches heavily rely on painstakingly customized turbulence indexes, which are less effective in dynamic and complex weather conditions. The recent availability of high-resolution weather data and turbulence records allows more accurate forecasting of the turbulence in a data-driven way. However, it is a non-trivial task for developing a machine learning based turbulence forecasting system due to two challenges: (1) Complex spatio-temporal correlations, turbulence is caused by air movement with complex spatio-temporal patterns, (2) Label scarcity, very limited turbulence labels can be obtained. To this end, in this paper, we develop a unified semi-supervised framework, T^2-Net, to address the above challenges. Specifically, we first build an encoder-decoder paradigm based on the convolutional LSTM to model the spatio-temporal correlations. Then, to tackle the label scarcity problem, we propose a novel Dual Label Guessing method to take advantage of massive unlabeled turbulence data. It integrates complementary signals from the main *T*urbulence Forecasting task and the auxiliary *T*urbulence Detection task to generate pseudo-labels, which are dynamically utilized as additional training data. Finally, extensive experimental results on a real-world turbulence dataset validate the superiority of our method on turbulence forecasting.**

***Index Terms*—turbulence forecasting, semi-supervised learning, spatio-temporal modeling**

I. Introduction

Turbulence is the leading cause of injuries to airline passengers and causes huge loss for airline companies. According to the U.S. Federal Aviation Administration (FAA), from 1980 through 2008, U.S. air carriers had 234 turbulence accidents, resulting in 298 serious injuries and three fatalities. More lately, since 2009, more than 340 turbulence related injuries have been reported[1]. The consequent economic losses are enormous. A vice president of one major air carrier once estimated that it pays out "tens of millions per year" for customer injuries, and loses about 7000 days in employee injury-related disabilities [1]. If turbulence can be forecasted accurately so that airlines can reroute ahead, then injuries and property damage can be averted, even lives can be saved.

Despite the benefits, turbulence has been difficult to forecast for being a "microscale" phenomenon. In the atmosphere, turbulent "eddies" vary in size, from hundreds of kilometers down to centimeters. But aircraft bumpiness is most pronounced when the turbulent eddies and aircraft are similar in size. It is impossible to directly forecast atmospheric motion at this scale, now or even in the foreseeable future [1]. Fortunately, most of the energy associated with turbulent eddies on this scale cascade down from the larger scales of atmospheric motion [2, 3, 4], and these larger scales may be resolved by Numerical Weather Prediction (NWP) models. Based on NWP, a variety of *turbulence indexes*, derived from basic weather features, are proposed by meteorologists to estimate the probability of turbulence occurrence [1]. While most existing turbulence forecasting methods rely on turbulence indexes, however, we observe that solely using manually crafted features is usually suboptimal, yielding unsatisfactory accuracy. Moreover, turbulence indexes have poor generalization power to adapt to new data, especially can not handle more complex situations such as climate change.

On the other hand, prior work has shown that a related problem, i.e., weather forecasting can be solved in a more effective and automatic way leveraging deep learning [5, 6], whereas, researches on applying advanced machine learning models to turbulence forecasting still remain few. To this end, we make the very first attempt to leverage deep learning for turbulence forecasting, using turbulence events recorded by pilot reports as ground truth labels. Nevertheless, we find two inevitable challenges impeding us from building an effective turbulence forecasting system:

- **Complex spatio-temporal correlations**. Turbulence is in nature a *spatio-temporal* phenomenon of air movements. It may occur as a result of various conditions, such as proximity to the jet stream or mountain waves. These conditions can actually be captured by certain combinations of meteorological features of the surrounding area and adjacent time slots. Most existing approaches only consider the static features of the target area but neglect the spatio-temporal features of surrounding areas.
- **Label scarcity.** Under the paradigm of supervised learning, a large number of turbulence labels are needed to provide signals for training a statistical forecasting model. However, the turbulence label is very *scarce* in the real-world because: (i) turbulence is a rare and anomaly event, (ii) it can only be recorded when there is a pilot happens to pass by Data with

*Corresponding author.
[1]https://www.faa.gov/news/fact_sheets

scarce labels, largely limits the power of machine learning.

To address the above challenges, we present a unified semi-supervised deep learning framework for turbulence forecasting, namely, T^2-Net. T^2-Net consists of two modules, i.e., a turbulence forecasting model and a turbulence detection model, which are co-trained in a semi-supervised manner. The forecasting model is built upon ConvLSTM to learn the complex spatio-temporal patterns for causing turbulence automatically. To take advantage of massive unlabeled data and alleviate the label scarcity issue, we propose a novel Dual Label Guessing (DLG) method for data augmentation. In DLG, we introduce an auxiliary task, **T**urbulence Detection, and employ 3D-CNN for this task. T^2-Net integrates complementary signals from the two tasks to generate more robust pseudo-labels, which are then utilized as additional data for better generalization ability. Finally, we carry out extensive experiments on a real-world dataset to evaluate our model. Results show that T^2-Net outperforms strong baseline methods in terms of all evaluation metrics (Accuracy, Weighted-Precision, Weighted-Recall, Weighted-F1). Hence the proposed approach can greatly alleviate the problem of spatio-temporal correlation modeling and label scarcity on turbulence forecasting.

II. Preliminary

In this section, we introduce several essential preliminaries. First, we give the problem formulation of turbulence forecasting as well as the auxiliary task, turbulence detection. Then we elaborate the features and labels used in our framework.

A. Turbulence Forecasting

We formulate the turbulence forecasting problem as a *sequence to sequence multi-class classification* problem. That is, given the historical feature cubes (each cube representing a grid-based 3D region) at previous time slots, $\boldsymbol{X_1}, \boldsymbol{X_2}, ..., \boldsymbol{X_n} \in \mathbb{R}^{\mathcal{L}\times\mathcal{W}\times\mathcal{H}\times\mathcal{C}}$, it aims to predict the turbulence levels of all grids in this 3D region at next few time slots, i.e., $\boldsymbol{Y_{n+1}}, \boldsymbol{Y_{n+2}}, ..., \boldsymbol{Y_{n+p}} \in \mathbb{R}^{\mathcal{L}\times\mathcal{W}\times\mathcal{H}\times 4}$. $\mathcal{L}\times\mathcal{W}\times\mathcal{H}$ indicates the size (number of grids) of the 3D region, C is the number of channels/features per grid, and 4 denotes the number of turbulence classes. Each time slot could be, for example, an hour, 3 hours, or a day. Let $\boldsymbol{X} = [\boldsymbol{X_1}, \boldsymbol{X_2}, ..., \boldsymbol{X_n}]$, $\boldsymbol{Y} = [\boldsymbol{Y_{n+1}}, \boldsymbol{Y_{n+2}}, ..., \boldsymbol{Y_{n+p}}]$, we aim to train a statistical model $\mathcal{F}(\cdot;\theta_{TFN})$, that, given $\boldsymbol{X}$, yields a forecast sequence $\boldsymbol{P_{TFN}}$ fitting $\boldsymbol{Y}$:

$$\boldsymbol{P_{TFN}} = \mathcal{F}(\boldsymbol{X};\theta_{TFN}) \tag{1}$$

In this paper, we set $\mathcal{L}\times\mathcal{W}\times\mathcal{H} = 10\times 10\times 5$ for computation efficiency and flexibility[2]. We choose one hour as the length of a time slot, in other words, we use the previous n hours' feature cubes to forecast the hourly turbulence level of next p hours.

[2]The receptive field of $10\times 10\times 5$ is large enough since each grid has the size of 13km, and turbulence "eddies" are normally smaller than 100km [1].

TABLE I: Raw features and turbulence indexes

Notation	Name	Unit
v_U	U component of wind	ms^{-1}
v_V	V component of wind	ms^{-1}
T	Temperature	K
H	Relative humidity	%
V	Vertical velocity	Pas^{-1}
P	Pressure	Pa
Ri	Richardson Number	-
CP	Colson Panofsky Index	kt^2
$TI1$	Ellrod Indices	s^{-2}
$\lvert v\rvert$	Wind Speed	ms^{-1}
$\lvert\bigtriangledown_H T\rvert$	Horizontal Temperature Gradient	Km^{-1}
$\lvert v\rvert DEF$	MOS CAT Probability Predictor	ms^{-2}

B. Turbulence Detection

Turbulence detection is a similar task to forecasting which serves as an auxiliary task in T^2-Net. Given the NWP forecasted feature cube of a time slot i, i.e., $\boldsymbol{X_i} \in \mathbb{R}^{\mathcal{L}\times\mathcal{W}\times\mathcal{H}\times\mathcal{C}}$, turbulence detection aims to predict turbulence conditions of all grids in this 3D region at the same time slot, i.e., $\boldsymbol{Y_i} \in \mathbb{R}^{\mathcal{L}\times\mathcal{W}\times\mathcal{H}\times 4}$. In this task, we aim to train a statistical model $\mathcal{F}(\cdot;\theta_{TDN})$, that, given $\boldsymbol{X_i}$, return detection result $\boldsymbol{P_{i,TDN}}$ fitting $\boldsymbol{Y_i}$:

$$\boldsymbol{P}_{i,TDN} = \mathcal{F}(\boldsymbol{X_i};\theta_{TDN}) \tag{2}$$

The detection task differs from forecasting task in two ways: (1) *Synchroneity*, i.e., its features are forecasted based on NWP models and synchronized with the turbulence labels. It aims to detect future turbulence using future features while forecasting aims to predict future turbulence using past features. (2) *Static*, it is also easier since it only predicts one step at one time. These two tasks share the same target but have different input features and hold different properties. We utilize both turbulence forecasting and detection to provide complementary guidance for the pseudo-label generation.

C. Features

In each grid of a feature cube (i.e., $\boldsymbol{X_i}$), we fill it with 12 relevant features (thus C=12) as shown in Table I. The first 6 of them are *raw weather features* while the rest 6 are *turbulence indexes* invented by meteorologists. Raw features such as temperature, wind component, and pressure can be considered as fundamental features and certain combinations of these features in adjacent areas may contribute to the occurrence of turbulence. *Deep neural network* such as *convolutional neural network* is capable of learning such complex spatial correlations and it is essential to keep the raw features. We further apply 6 turbulence indexes as extra features to enhance the model capacity. Most of these indexes are proposed by previous meteorologists, usually adopted independently or integrated by a weighted sum [1] in existing turbulence forecasting systems. We regard them as prior knowledge and concatenate with raw features.

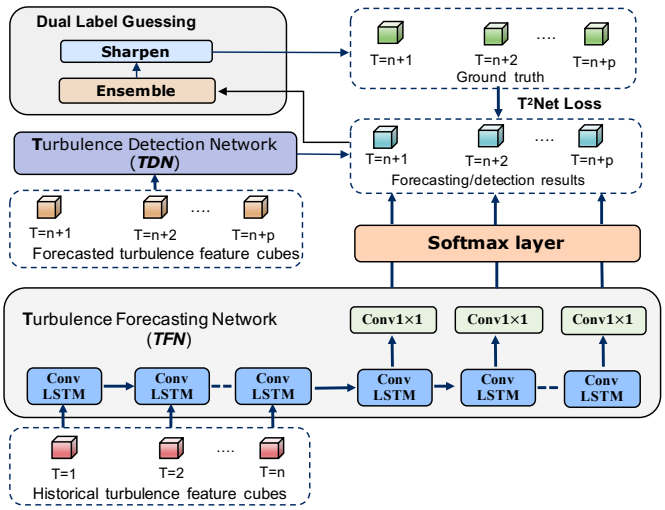


Fig. 1: The architecture of T^2-Net

D. Labels and the scarcity issue

We collect turbulence events data from online pilot reports, each is labeled with a severity level. There are four levels in our data, i.e., Negative, Light, Moderate, and Severe. After gathering the feature data and label data, we align them by time and space. Details are provided in Section 4. According to our statistics, at each hour, there are only 0.05% grids of North American air space are labeled with a turbulence level while 99.95% are unknown. Consequently, we have to mask these unlabeled grids during training to bypass the backpropagation of their gradients. This leads to less training signals available, making it hard for the network to be trained sufficiently.

III. Methodology

In this section, we introduce the details of our proposed turbulence forecasting framework, T^2-Net. As shown in Figure 1, T^2-Net mainly consists of a ***T**urbulence Forecasting Network* (**T**FN) and a ***T**urbulence Detection Network* (**T**DN). TFN serves for the main task, i.e., forecasting task, while TDN serves for the auxiliary task, i.e., turbulence detection. Based on the predictions of TFN and TDN, a novel ***Dual Label Guessing*** approach is proposed to generate more robust pseudo-labels as additional training data.

A. Turbulence Forecasting Network

TFN is designed on top of the basic ConvLSTM [6] architecture to model the complex spatio-temporal correlations among different spatial grids. ConvLSTM is a variation of LSTM which extends basic LSTM cell by replacing the fully connected layer with convolution operation in the internal transitions. As shown in Figure 1, TFN consists of two ConvLSTMs, serving as the encoder and decoder respectively. The encoder takes a sequence of 4D tensors as input, $\boldsymbol{X_1}, \boldsymbol{X_2}, ..., \boldsymbol{X_n} \in \mathbb{R}^{\mathcal{L}\times\mathcal{W}\times\mathcal{H}\times\mathcal{C}}$, i.e., the historical turbulence feature cubes of time slots $1, ..., n$. The decoder takes the last hidden state of the encoder as the initial hidden state, and uses *teacher forcing* [7] (use previous ground truth $\boldsymbol{Y_{j-1}}$ as the next input to the decoder) to generate a sequence of features corresponding to the forecasting time slots $n+1, ..., n+p$. The decoder's outputs are then fed into to a Conv1 $\times$ 1 block followed with a Softmax layer to produce the forecasted turbulence levels $\boldsymbol{P_{n+1}}, \boldsymbol{P_{n+2}}, ..., \boldsymbol{P_{n+p}} \in \mathbb{R}^{\mathcal{L}\times\mathcal{W}\times\mathcal{H}\times 4}$. The process of TFN can be summarized as:

$$\boldsymbol{h_i^{enc}}, \boldsymbol{o_i^{enc}} = \text{ConvLSTM}^{enc}\big(\boldsymbol{X_i}, \boldsymbol{h_{i-1}^{enc}}\big), i \in [1, n]$$

$$\boldsymbol{h_j^{dec}}, \boldsymbol{o_j^{dec}} = \text{ConvLSTM}^{dec}\big(\boldsymbol{Y_{j-1}}, \boldsymbol{h_{j-1}^{dec}}\big), j \in [n+1, n+p]$$

$$\boldsymbol{P_j} = \text{Softmax}\big(\text{Conv1} \times 1(\boldsymbol{o_j^{dec}})\big), j \in [n+1, n+p]$$

B. Turbulence Detection Network

The Turbulence Detection Network (TDN) employs *Convolutional Neural Network* to extract spatial correlations and detect the turbulence levels. The input to TDN is the NWP forecasted turbulence feature cube X_i at time slot i, and the output is the detected turbulence level cube $P_i \in \mathbb{R}^{\mathcal{L}\times\mathcal{W}\times\mathcal{H}\times 4}$ at the same time slot. TDN can be summarized as:

$$\text{Conv}\big(\boldsymbol{X_i}, 1\big) = f_1\big(\boldsymbol{X_i} \circledast \boldsymbol{W_1} + b_1\big), \tag{3}$$

$$\text{Conv}\big(\boldsymbol{X_i}, l\big) = f_l\big(\text{Conv}(\boldsymbol{X_i}, l-1) \circledast \boldsymbol{W_l} + b_l\big), \tag{4}$$

$$\boldsymbol{P_i} = \text{Softmax}\big(\text{Conv}(\boldsymbol{X_i}, l)\big), i \in [n+1, n+p], \tag{5}$$

where l denotes the l-th layer, f_l denotes the activation function of l-th layer, "$\circledast$" denotes the 3D-convolution operator.

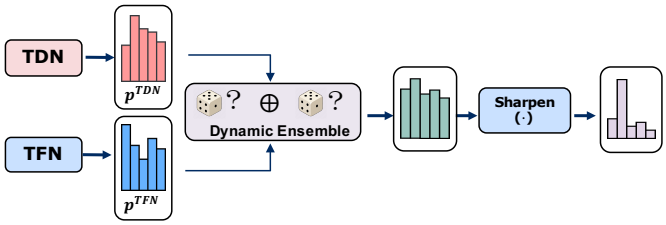


Fig. 2: Diagram of dual label guessing

C. Dual Label Guessing

To mitigate the label scarcity issue, we propose *Dual Label Guessing* (DLG), as illustrated in Figure 2. During training, DLG will generate pseudo-labels for those unlabeled grids so that we can obtain additional training data. To highlight, DLG differs from existing "label-guessing" methods [8, 9] in two ways:

- ***Complementary Dual Semi-supervised Signals.*** Instead of *single source* inference, our method leverages *dual source* signals from two related but different tasks. DLG combines the predictions from TDN and TFN, protecting each other from their individual errors/bias, thus getting more robust to generate high-quality pseudo-labels.
- ***Soft Labels.*** Instead of the *hard label* in other approaches like "pseudo-labeling" [8] which takes the class with the highest probability and produce a one-hot label, we produce *soft label* via a "sharpening" function [9], yielding a class distribution. The soft label is smoother and more error-tolerant compared with hard label.

*1) **Dynamic Ensemble of TDN and TFN**:* In our Dual Label Guessing, we first propose a novel *Dynamic Ensemble* method to fuse the predictions of TFN and TDN grid by grid, the combined prediction is defined as:

$$\boldsymbol{p} = \frac{\Psi(\boldsymbol{p}^{TDN}, \boldsymbol{p}^{TFN}, \tau(t)) \oplus \Psi(\boldsymbol{p}^{TDN}, \boldsymbol{p}^{TFN}, \tau(t))}{2} \tag{6}$$

where $\boldsymbol{p}^{TDN}, \boldsymbol{p}^{TFN} \in \mathbb{R}^4$ are output vectors of a single grid predicted by TDN and TFN respectively, in which each element represents the probability of each turbulence class. $\oplus$ denotes element-wise addition. Ψ denotes the ***binary sampling***. To be noted, two $\Psi(\boldsymbol{p}^{TDN}, \boldsymbol{p}^{TFN}, \tau(t))$ in the equation are different samples and the sampling function Ψ is defined as:

$$\Psi(\boldsymbol{p}^{TDN}, \boldsymbol{p}^{TFN}, \tau(t)) = \begin{cases} \boldsymbol{p}^{TDN}, & \text{if } r(t) > \tau(t) \\ \boldsymbol{p}^{TFN}, & \text{if } r(t) <= \tau(t) \end{cases} \tag{7}$$

$r(t)$ above is a *pseudorandom number* between $[0, 1]$ with t as the seed. $\tau(t)$ is a *dynamic coefficient* controlling the probability of drawing $\boldsymbol{p}^{TDN}$ or $\boldsymbol{p}^{TFN}$, i.e., *relative importance* of TDN and TFN, $\tau(t)$ is defined as a piece-wise function:

$$\tau(t) = \begin{cases} 0 & t < T_1 \\ \frac{t-T_1}{T_2-T_1}\beta & T_1 < t < T_2 \\ \beta & t > T_2 \end{cases} \tag{8}$$

where t is the number of epochs, T_1, T_2 and β are hyper-parameters. The design of $\tau(t)$ follows the intuitions: at the beginning of training, TDN shall have a higher probability (in the first stage, $1 - \tau(0) = 1$ makes TDN 100% to be chosen), because TDN is pre-trained, predicting more accurately than TFN. As the iteration t increases gradually, TDN's probability should decrease and TFN's increases since TFN's accuracy is growing. Finally, the binary sampling probability stabilizes at some balancing point $\beta \in (0, 1]$.

*2) **Soft Labels**:* After getting the ensembled prediction p, to obtain the pseudo-label, we further apply a sharpening function to minimize the *entropy* of the label distribution, which is defined as:

$$\text{Sharpen}(\boldsymbol{p}, T)_i := \boldsymbol{p}[i]^{\frac{1}{T}} \Big/ \sum_{j=1}^{4} \boldsymbol{p}[j]^{\frac{1}{T}} \tag{9}$$

where $\boldsymbol{p}[i]$ is the i-th element of $\boldsymbol{p}$, T is a hyper-parameter to adjust the "temperature" of this categorical distribution. **Sharpen**$(\boldsymbol{p}, T)$ first calculates the T-th power of each elements and then based on which performs a normalization. When $T \to 0$, the result will approach a one-hot distribution.

D. Loss Function

The loss function of our T^2-Net includes two parts: (1) $\mathcal{L}_s$, the supervised part for the labeled grids, (2) $\mathcal{L}_u$, the unsupervised part for the grids with pseudo-labels.

$$\mathcal{L} = \mathcal{L}_s + \lambda \mathcal{L}_u \tag{10}$$

where $\lambda \in [0, 1]$ is a hyperparameter controlling the weight of unsupervised loss. For $\mathcal{L}_s$, we adopt cross-entropy, and for $\mathcal{L}_u$, we employ the L2 distance between model predictions and pseudo labels.

TABLE II: Overall performance of T^2-Net and baselines.

Method	Accuracy	Weighted		
		P	R	F1
TBI (Turbulence indexes)	0.428	0.368	0.428	0.382
Multinomial LR	0.434	0.355	0.434	0.385
MLP (3-layers)	0.449	0.393	0.449	0.355
GBDT (100 trees)	0.440	0.370	0.440	0.341
Attentional LSTM	0.489	0.378	0.489	0.426
CNN (kernel size=3)	0.491	0.437	0.491	0.370
ConvLSTM	0.571	0.548	0.581	0.518
Hetero-ConvLSTM	0.580	0.536	0.580	0.520
Pseudo-labeling	0.591	0.536	0.600	0.536
Mixmatch	0.600	0.546	0.580	0.540
T^2-Net	**0.623**	**0.551**	**0.614**	**0.548**

IV. Experiments

In this section, we first introduce the data and settings of our experiments. Then we evaluate the performances of the proposed forecasting model on different forecasting lengths compared with several state-of-the-art baselines. Lastly, we show the parameter sensitivity analysis.

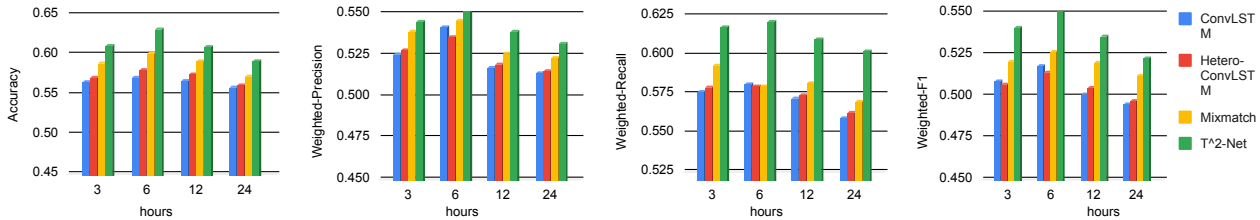
A. Experimental Settings

Data Preprocessing: All the data we use in the experiment are publicly available. We first obtain 30 days of weather data (from 20190601 to 20190630) from National Oceanic and Atmospheric Administration (NOAA) [3], then obtain 30 days of turbulence report data (the same period as the weather data) from Iowa Environmental Mesonet (IEM) [4]. The weather data is generated hourly on a 13-km (8-mile) resolution horizontal grid, with 451×337 grids in total, representing the North American region. In vertical direction, there are 36 different geopotential heights. We treat the whole data as a $451 \times 337 \times 36$ cube and use a sliding window of size $10 \times 10 \times 5$ to generate the hourly *feature cubes* (each grid contains the raw weather features and the turbulence indexes). For the turbulence report, each report contains the coordinate, geopotential height, time, and level of the turbulence, and we use the same way to generate the hourly *label cubes* (each grid contains the turbulence level). We filter out the cubes NOT in the "cruising altitude" (31000 to 38000 feet) since most pilot reports are recorded among these heights. Finally, we use a sliding window of $n+p$ hours, adopt the first n hours' feature cubes as the input sequence and the next p hours' label cubes as the output sequence to generate the training data. In experiments, we investigated different n and p, specifically, $n = p = \{3, 6, 12, 24\}$. We further randomly split the data into training/validation/testing set with the ratio of 6:2:2.

Evaluation Metrics: We adopt several standard multi-class classification metrics to evaluate all the models: (1) Accuracy, (2) Weighted-Precision, (3) Weighted-Recall, (4) Weighted-F1. We calculate these metrics by averaging them throughout all the labeled grids in different time steps and skip the grids labeled with "unknown".

[3] ftp://nomads.ncdc.noaa.gov/RUC/13km/201906/

[4] http://mesonet.agron.iastate.edu/request/gis/pireps.php



(a) Accuracy (b) Weighted-Precision (c) Weighted-Recall (d) Weighted-F1

Fig. 3: Performance comparison of different forecasting lengths.

Parameter Configuration: For our model and all the baseline methods, we obtain the optimal parameters on the validation set using early stopping. For TFN, we adopt 1-layer ConvLSTM with $3 \times 3 \times 3$ kernel for both encoder and decoder, using Sigmoid as activation function. For TDN, we adopt 3-layer CNN with $3 \times 3 \times 3$, $3 \times 3 \times 3$ and $5 \times 5 \times 3$ kernels respectively, using Relu as inner activation function. The optimal hyperparameters of the rest part are $\beta = 0.6, T_1 = 5, T_2 = 15, T = 0.5, \lambda = 0.4$.

Baseline Methods: To systematically investigate the performance of modern machine learning methods on turbulence forecasting, we compare our proposed T^2-Net with 3 categories of baselines. (1) *Tubulence indexes* (**TBI**) [1], an integrated approach which combines multiple turbulence indexes to forecast turbulence. (2) *Supervised learning methods*: a number of supervised machine learning methods are examined: Multinomial Logistic Regression (**Multinomial LR**), Multi-layer Perceptron (**MLP**), Gradient Boosting Decision Tree (**GBDT**) [10], **Attentional LSTM**, Convolutional Neural Network (**CNN**), **ConvLSTM** [6] and **Hetero-ConvLSTM** [11]. We test all these methods using the same base features as our model, i.e., the 6 raw weather features and 6 turbulence indexes. (3) *Semi-supervised learning (SSL) methods*: we also compare with several state-of-the-art semi-supervised learning methods since abundant unlabeled data exists in the turbulence forecasting task and T^2-Net is also semi-supervised. **Pseudo-labeling** [8], a simple yet effective SSL method which retrains the model with pseudo-labels predicted by the model itself. **Mixmatch** [9], a recent holistic approach which unifies several dominant SSL methods and achieves state-of-the-art results on 4 benchmark datasets. To ensure a fair comparison, we apply the same base model (ConvLSTM) for these SSL methods.

B. Overall Results on Turbulence Forecasting

We first present the overall performance of all baselines and T^2-Net when $n = p = 6$. The results are presented in Table II. MLP and GBDT perform better than TBI and Multinomial LR because they integrate nonlinearity. By taking temporal correlation and spatial correlation into consideration, Attentional LSTM and CNN achieve better accuracy than those not. ConvLSTM further improves the performance because of modeling spatial and temporal correlation simultaneously. Hetero-ConvLSTM achieves the best accuracy and Weighted F1 among supervised methods for it predicts based on an ensemble of multiple ConvLSTMs of different geographical areas. However, Hetero-ConvLSTM has an efficiency issue for training many ConvLSTMs at the same time. For the semi-supervised learning baselines, all of them beat the supervised methods, verifying the practicability of taking advantage of unlabeled data. Among them, Mixmatch achieves the best accuracy, Weighted F1, Weighted Precision, and thus proves that *soft pseudo-label* is better than *hard pseudo-label*. T^2-Net achieves the best performance with the highest Accuracy, Weighted Precision/Recall/F1. This indicates that on turbulence forecasting task, T^2-Net has superior ability to model the complex spatio-temporal relation as well as utilizing the abundant unlabeled data to enhance training.

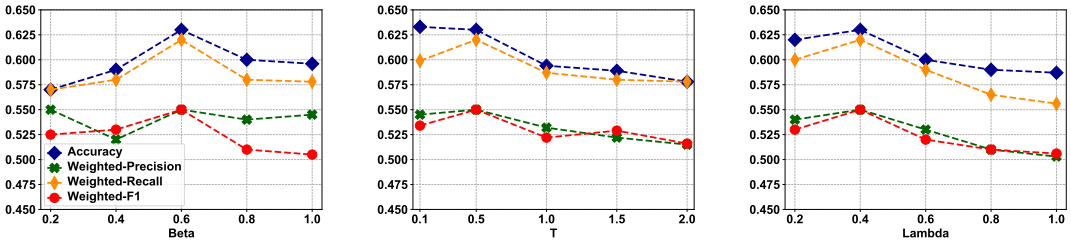
We also report the overall performance of T^2-Net and several representative baselines (ConvLSTM, Hetero-ConvLSTM, and Mixmatch) for different forecasting lengths ($n = p = 3, 6, 12, 24$). As shown in Figure 3, we can observe that T^2-Net achieves the best performance on different time lengths. However, an interesting phenomenon brought to our attention is that when n, p increases (>6), there is a drop in performance for all the baseline models and T^2-Net. We attribute this to the increased complexity and "gradient vanishing" problem in long-sequence prediction. However, performance on $n, p = 6$ is better than $n, p = 3$, this is because the model benefits more from the increased input feature length when the time length changes from 3 to 6.

C. Parameter Sensitivity Analysis

Figure 4 presents the sensitivity analysis of the key parameters of our Dual Label Guessing and loss function, i.e., β, T, and λ. We obtained the sensitivity curve of each parameter by fixing the rest using their optimal values. We can observe that the best performance is achieved when $\beta = 0.6$, this indicates TFN is more important to the final result. Besides, $\lambda = 0.4$ achieves the best results indicating that there is a trade-off of utilizing the unlabeled data. We can also observe that the performance is relatively stable as the parameters change, thus proves the robustness of T^2-Net.

V. Related Work

Traditional turbulence forecasting approaches mainly focus on devising various turbulence indexes [12, 13, 14]. Sharman et al. reviewed 13 turbulence indexes and proposed an integrated method combining these features to achieve further improvements [1]. However, as validated in our experiments, methods solely rely on turbulence indexes can not achieve



(a) β (b) T (c) λ

Fig. 4: Parameter sensitivity analysis.

satisfactory performance. Our method takes advantage of both turbulence indexes and the power of deep learning to achieve superior performance. Recent years have witnessed a growing interest in applying machine learning to a variety of spatio-temporal problems [6, 15, 16]. Most of them adopt ConvLSTM to model the spatio-temporal correlations without facing the label scarcity issue. Particularly, we propose a novel semi-supervised approach to tackle the scarcity issue considering by taking complementary turbulence signals into account. More recently, some work tries to incorporate physical principles with deep learning to facilitate turbulent flow modeling, however, they are not directly tackling turbulence forecasting, but simulating and predicting certain turbulence variables, e.g., Wang et al. predicts the velocity fields [17]. To our best knowledge, our work is the first systematic machine learning study, that directly forecasts the occurrence of flight turbulence, using sparse turbulence labels extracted from pilot reports as supervision.

VI. Conclusion

In this paper, we developed a data-driven framework for turbulence forecasting. Specifically, we first built an encoder-decoder paradigm based on ConvLSTM to model the spatio-temporal correlations. Then, to address the label scarcity issue, we proposed a novel Dual Label Guessing method, which integrated complementary signals from the main task of **T**urbulence Forecasting and the auxiliary task of **T**urbulence Detection to generate pseudo-labels. Finally, we conducted extensive experiments on a real-world dataset which showed that the proposed approach can greatly alleviate the problem of spatio-temporal correlation modeling as well as label scarcity on turbulence forecasting.

References

[1] R. Sharman, C. Tebaldi, G. Wiener, and J. Wolff, "An integrated approach to mid-and upper-level turbulence forecasting," *Weather and forecasting*, 2006.

[2] J. A. Dutton and H. A. Panofsky, "Clear air turbulence: A mystery may be unfolding," *Science*, 1970.

[3] K. K. Tung and W. W. Orlando, "The k- 3 and k-5/3 energy spectrum of atmospheric turbulence: Quasi-geostrophic two-level model simulation," *Journal of the atmospheric sciences*, 2003.

[4] J. N. Koshyk and K. Hamilton, "The horizontal kinetic energy spectrum and spectral budget simulated by a high-resolution troposphere–stratosphere–mesosphere gcm," *Journal of the atmospheric sciences*, 2001.

[5] H.-P. Hsieh, S.-D. Lin, and Y. Zheng, "Inferring air quality for station location recommendation based on urban big data," in *SIGKDD*, 2015.

[6] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *NIPS*, 2015.

[7] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[8] D.-H. Lee, "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," in *ICML Workshop*, 2013.

[9] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. Raffel, "Mixmatch: A holistic approach to semi-supervised learning," *arXiv preprint arXiv:1905.02249*, 2019.

[10] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, 2001.

[11] Z. Yuan, X. Zhou, and T. Yang, "Hetero-convlstm: A deep learning approach to traffic accident prediction on heterogeneous spatio-temporal data," in *SIGKDD*, 2018.

[12] D. Colson and H. Panofsky, "An index of clear air turbulence," *Quarterly Journal of the Royal Meteorological Society*, 1965.

[13] M. Dutton, "Probability forecasts of clear-air turbulence based on numerical model output," 1980.

[14] G. P. Ellrod and D. I. Knapp, "An objective clear-air turbulence forecasting technique: Verification and operational use," *Weather and Forecasting*, 1992.

[15] A. Karpatne, Z. Jiang, R. R. Vatsavai, S. Shekhar, and V. Kumar, "Monitoring land-cover changes: A machine-learning perspective," *IEEE Geoscience and Remote Sensing Magazine*, 2016.

[16] Z. Yuan, H. Liu, Y. Liu, D. Zhang, F. Yi, N. Zhu, and H. Xiong, "Spatio-temporal dual graph attention network for query-poi matching," in *Proceedings of the 43rd International ACM SIGIR*, pp. 629–638, 2020.

[17] R. Wang, K. Kashinath, M. Mustafa, A. Albert, and R. Yu, "Towards physics-informed deep learning for turbulent flow prediction," *arXiv preprint arXiv:1911.08655*, 2019.