Master's Thesis

Detecting phenological transition dates of vegetation based on multiple deep learning models

Zhaoyang Cheng
zyclark.cheng@gmail.com

Thesis committee:

Prof. dr. ir. M.J.T. Reinders
Dr. J.C. van Gemert
Dr. S. Picek
Dr. S. Khademi

August 2018

TU Delft

Abstract

Vegetation phenology is the interaction between vegetation activities and ecosystem. Accurate monitoring of vegetation phenology is required to build models and enhance the understanding of the relationship between creatures and climate-environment. PhenoCam is a ground-level, webcam based images database recording the growing of various vegetations, PhenoCam and multiple modeling methods have been utilized to study vegetation phenology since 2000s. In this paper, it first time the deep learning models are applied to detect the phenological transition dates of vegetation. Four different deep learning models: Convolution Neural Network (CNN), Siamese Network, 3-D Fully Convolution Neural Network (FCN) and Regression Network are used to study the vegetation phenology, based on these approaches, the transition dates of vegetation activities within annual time can be determined from webcam-based images, some of these deep learning methods are more accurate than traditional modeling method in detecting the transition dates.

1. Introduction

The vegetation dynamics in ecosystems largely reflect the response of the biosphere to dynamics of the climate [1][2][3]. Analysis on vegetation dynamics have provided important record of how vegetations have responded to climate change, however, observation by human on long-term vegetation dynamics is laborious and time-consuming. Accurate and automatic monitoring of vegetation dynamics is therefore an important work for researchers to investigate.
PhenoCam is a database consisting of inter-annual digital images of various vegetations throughout North America. Because of the consistent record of phenology of different vegetations, PhenoCam has played an important role in building model to monitor vegetation dynamics at regional scales. In the study of building models based on PhenoCam, a key challenge is determining how the phenology derived from webcam based images relate to biological events that a human observer would recognize. In the last decade, a number of different methods have been developed to determine the timing of vegetation's greenup and senescence(i.e. the start and end of growing season of vegetation) which are called transition dates of vegetations, most methods extract the vegetation index (VI) that calculates a specific feature of vegetation, e.g. green chromatic coordinate (GCC) and red chromatic coordinate (RCC), and build sigmoid-based model to fit the vegetation index, the curve of fitted model can largely reflect the growing of vegetation. Instead of choosing a specfic vegetation index, utilizing deep learning to automatically extract the most distinguishable features is more convenient and accurate.
In this thesis, my research topic is how to apply deep learning methods to detect transition dates of vegetation based on images of vegetations. The research data includes the data of four sites: (1) Queens; (2) Bartlettir; (3) Umichbiological; (4) Harvard, in PhenoCam database.

1.1. Monitoring phenology using PhenoCam

Vegetations have many key phenological phases, e.g., the date when leaves start becoming green, the date when flowers come out, etc.. In this thesis, four basic phenological activities are investigated: (1) the dates when the majority of trees started leafing out, its phenological meaning is the onset of photosynthetic [4]; (2) the dates when green leaf area is maximum, it phenological meaning relate to the end of spring; (3) the dates when the canopy first started to change color in the fall, which means the onset of senescence; (4) the dates when the majority of trees had lost all leaves, its phenological meaning corresponds to the start of dormancy. Field based ecological studies have demonstrated that vegetation phenology tends to follow relatively well-defined temporal patterns [5]. The dates when leaves green up tend to be followed by a period of rapid growth, followed by a relatively stable period of maximum leaf area, after the brightest color phase in fall, leaf area decreases dramatically. The vegetation index(VI) used to quantify phenolog-

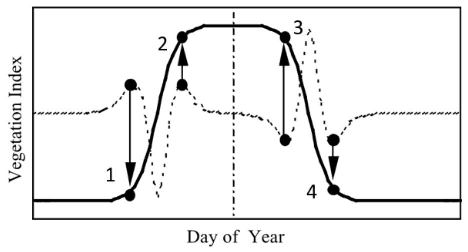

Figure 1. A synoptic figure [5] of how transition dates are calculated by the extremas in the rate of change of curvature, the solid line is an idealized curve of VI with respect to day of year, the dashed line is the rate of change of VI curve.

ical status of the vegetation over time is usually green chromatic coordinate(GCC), researchers can fit logistic function to VI from PhenoCam and find key transition points (usually the extremas) which are identified as transition dates, however, the simple logistic function is not accurate enough in detecting the transition dates because of the complexity of temporal and spatial information of PhenoCam data. They therefore propose some modified sigmoid methods based on empirical equations.

2. Related Work

In this section, I first introduce some sigmoid-based models that have been successfully applied in detecting the transtion dates of vegetation, then I introduce previous works that study phenology through deep learning.

$$GCC = \frac{Green}{Green + Red + Blue} \tag{1}$$

The vegetaion index used in forest phenology is green chromatic coordinate(GCC), equation 1 is a non-linear transformation of the camera's measured green digital numbers to values representing the proportion of the greenness measured, GCC measures the changing levels of green pigmentation in vegetation. An advantage of using GCC is to reduce the influence of differences in scene illumination between images, literature [6] found that weather-induced changes in scene illumination are largely suppressed when using GCC, it is especially helpful when applying GCC index to PhenoCam data because illumination is often different in morning and afternoon, illumination also changes quickly with different weather conditions in PhenCam datasets.

2.1. Simple sigmoid-based method

$$f_s(t) = \frac{c}{1 + exp(a + bt)} + d \tag{2}$$

Equantion 2 is the Simple Sigmoid moedl, which is widely used in phenology community [5][6][7]. $f_s(t)$ represents the model value of vegetation index, parameter a decides the time of decrease or increase, parameter b controls the rate of increase or decrease, parameter c is the amplitude of increase or decrease in vegetation index, parameter d defines the dormant season baseline value of vegetaion index.

2.2. Generalized sigmoid-based method

The Simple Sigmoid model does not work accurately enough in fitting the decreasing greenness in summer time, so Elmore *et al*. [8] propose a Double Sigmoid model,

$$f_{pv}(t) = m_1 + (m_2 - m_7 * t) * V \tag{3}$$
$$V = \frac{1}{1+exp((m_3-t)/m_4)} - \frac{1}{1+exp((m_5-t)/m_6)} \tag{4}$$

in equation 3, $f_{pv}(t)$ is photosynthetic vegetation fraction (FPV) stacked by day of year, $f_{pv}(t)$ is related to the pattern of leaf development and growing season stability. m_1 is the average FPV measured in winter, m_2 is the difference between FPV measured in summer, m_3 and m_4 control the shape of growth curve in summer time, m_5 and m_6 control the shape of growth curve in winter time, the parameter m_7 tunes the seasonal vegetation index, hereby makes this model more accurate. Recently a more flexible model is presented by [9], they introduce two additional parameters v_i, q_i in equation 6 , which allow more flexible rates of increase near the lower and upper asymptotes of the sigmoid-based function. This Generalized Sigmoid model can not only control the baseline value of vegetation index via a_1, but fit the greenness decrease in summer time which is a common phenomena in many sites via a_2 and b_2.

$$f(t) = a_1 * t + b_1 + (a_2 * t^2 + b_2 * t + c) * V \tag{5}$$
$$V = \frac{1}{[1+q_1 exp(-h_1(t-n_1))]^{v_1}} - \frac{1}{[1+q_2 exp(-h_2(t-n_2))]^{v_2}} \tag{6}$$

2.3. Local extremas and transition dates

For each sigmoid-based method, the local extrema in the rate of change of curvature is estimated as phenological transition date,

$$k = \frac{f''(t)}{(1 + (f'(t))^2)^{\frac{3}{2}}} \tag{7}$$

equation 7 [10] can be used to calculate the rate of change of curvature k. In Simple and Double sigmoid model, points where the k is largest and smallest in the seasonal transition phase are identified as transition dates, for example, in Figure 1, point 1 and 2 are estimated as the transition dates for event 1 (majority of trees leafing out) and event 2 (green leaves area reaches maximum) respectively, point 3 and 4 are identified as event 3 (leaves start changing color in fall) and event 4 (majority of trees lose leaves) respectively.
For Generalized Sigmoid model, the third extreme in the curvature change rate is used to detect the transition date of event 2. the transition date of event 1 is identified as the date corresponding to 10% amplitude between the dormant season and the values of vegetation index at event 2, the detection approach is similar in fall.

2.4. Deep learning and phenology

There is no previous work that applies deep learning to datasets from PhenoCam, while research usually apply deep learning model to plants classification by convolution neural network (CNN) and long short term memory network (LSTM)[11], there are also applications: classification of vegetation growing stages by CNN [12], corp yield estimation by CNN [13]. In these studies, every training image has a corresponding label, and test images are assigned labels at prediction stage, which is same with classification tasks that have been solved by CNN in other fields.
In the detection of transition date problem, transition date is only one day of year, thus, the visually annotated label of transition date is also one day of year, which means many samples do not have labels, it is difficult to perform classification with CNN, literature [14] inspires me to solve the transition detection problem by video shot boundary detection methods.
Considering the scale of PhenoCam database, deep learning has potential to solve phenological problems with PhenoCam, however, shortage of label and the format of label could be problems if I want to apply deep learning to PhenoCam dataset, as unsupervised learning is unsuitable in

solving phenological problems at present. To tackle above problems, I present several deep learning models in following sections.

3. Methods

Looking through the aforementioned sigmoid-based algorithms in a deep learning worker's perspective, I find that they all belong to unsupervised learning style, to acquire more accurate estimation of transition dates, I bring the human-annotated label in and build different neural network models to detect transition date in a supervised learning way, morever, once the trained model is built, it can be applied to data in coming years.
To define the phenological problem with deep learning methodologies , training data, label and test data should be defined at the beginning, I therefore regard data in year 2008 and 2009 as training data, and data in year 2010 as test data. The label is the day of year that transition event happens. The data is continuous time-series images. I propose four different methods to solve this problem in different perspectives, the 3-D fully convolution neural network and regression network exploit the temporal information of data, while convolution neural network and siamese network focus on using spatial information of images.

3.1. Classification by convolution neural network

In conventional classification task, the sample size of each class is basically same, and every traning sample has a corresponding label, however, in PhenoCam dataset, labeled samples is too few to allow us perform a balanced classification, I therefore utilize data augmentation to make the class size roughly balanced.

3.1.1 Data augmentation

I find the transition date is the most representive day during the transition period, but the transition period usually lasts few days, so at first stage I decide to 'expand' the transition dates label in a small scale, in other words, I assign same label to the former and later E days of labeled transition date, these days are also regarded as transition dates. By visually inspect, the candidates of E are [3, 4, 5, 6, 7], how to choose the best 'expanded index' E is another task to do, a naive classification model consisting of convolutional layers and fully connected layers is used to find the best E, the expanded four events are regarded as four classes, for the rest of images, they are 5-th class, let us call it the noise class hereafter, however, even having expanded the labeled four event classes, the images in noise class is still much more than those in other four classes, to make class size balanced, I use data augmentation to increase the number of images of four event classes, the first measure I take is horizontal flipping of image, horizontal flipping reserves the characteristics of vegetation images and increase the number of images of four classes by the factor of two. I also notice that image's brightness changes over time in each day due to different illumination and weather condition, for example, the images taken in the morning and images taken in the afternoon may look different because of different brightness, to suppress the weather-induced change of illumination, I use gamma correction[15] to generate one darker and one brighter image from every original image,

$$I_{out} = I_{in}^{\gamma} \tag{8}$$

in equation 8,I_{in} and I_{out} is the image before and after gamma correction, γ is the parameter to control the brightness of image, by setting γ 1.1 and 0.9, a darker and a brighter image is generated based on original image. I therefore have a new dataset three times the size of previous dataset. When using image fliping and gamma correction together, the new dataset is six times the size of original dataset.
The data augmentation is also applied to the noise class, by sampling images from the noise class, there are five class with balanced size. I use data in 2008 as training data, data in year 2009 as validation data and data in 2010 as test data to select the best 'expand index' N, in my experiments, N is chosen as 5, which means the labeled transition dates are not 1 day but 11 days.
After augmentation, the only problem is how to design the architecture of deep learning model, for this common classification task, a convolutional neural network (CNN) is used, CNN has achieved great success since the AlexNet in ImageNet Challenge 2012, more and more nets with improved architecture are developed [16][17][18], they are widely applied in classification, object detection, semantic segmentation, etc.. CNN-based solutions usually have much higher accuracy , they are replacing traditional machine learning algorithms in many fields. In my scenario, for each site there are about 600 images per class (300 for site Bartlettir), advance network, e.g. ResNet [18] or VGG [17] is not used here otherwise serious overfitting will occur and the accuracy would not be high.
The architecture is shown in Table 1. When designing the architecture of network, I follow the principle of designing VGG net[17], with a given receptive field(the effective area size of input image on which output depends), multiple stacked smaller size kernel is better than the one with a larger size kernel because more non-linear layers increase the depth of the network which enables it to learn more complex features, and at a lower cost. At the beginning of designing the network, I use seven convolution layers, when number of convolution layers is reduced to four, the model still yields good result, I therefore only use model with four convolution layers. The number of parameters of the net is

Layer	Input size	Kernel size
data	N 256 256 3	3 3 3 8
layer 1	N 256 256 8	8 3 3 8
pool 1	N 128 128 8	
layer 2	N 128 128 8	8 3 3 16
pool 2	N 64 64 16	
layer 3	N 64 64 16	16 3 3 16
pool 3	N 32 32 16	
layer 4	N 32 32 16	16 3 3 32
pool 4	N 16 16 32	
flaten layer	N 16×16×32	
fc layer1	N 50	
fc layer2	N 5	
softmax layer	N 5	

Table 1. Architecture of our classification net, fc layer means fully connected layer, we use max pooling [16] to retain the important image information and reduce the number of parameters of model.

more than the number of samples, to suppress overfitting, I use dropout [19] in fully connnected layer. At the softmax layer, each test image is assigned to a class, to find the transition date, I find all the dates of test images in one class, after discarding the earliest date and latest date, the averaged date is the predicted transition date for test dataset.

3.2. 3-D fully convolution neural network

In CNN-based classification method, the temporal information of PhenoCam dataset is ignored. Inspired by the methods used in video shot boundary detection [14], I treat the images dataset as a continuous video, such that the transition date can be found in the same way that shot boundary in video is found. 3-D means dimension width, height, and time, 3-D convolution neural network (CNN) takes video snippets as input, and the convolution kernel is a 3-D cube rather than a 2-D window. To reduce number of parameters, I use fully convolution network (FCN) [20], in other words, there is no fully connected layers in FCN. Figure 2 shows

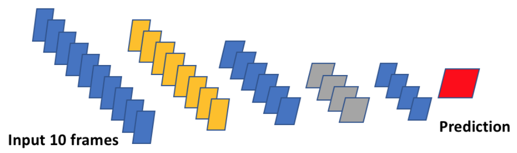

Figure 2. Illustration of 3-D FCN for shot boundary detection. each frame prediction is based on the context of N frames, in my setting, N is 10, the frame 6 of input 10 frames is predicted as shot boundary or not.

how the input frames relate to the prediction by 3-D FCN when batch size is 1. If the frame 6 of input 10 frames is a shot boundary, the input snippet will be annotated as 1, otherwise this snippet is labeled as 0.

Layer	Input size	Kernel size	stride
data	N 10 64 64 3	3 5 5 3 16	1 1 2 2 1
layer 1	N 8 30 30 16	3 3 3 16 24	1 1 2 2 1
layer 2	N 6 14 14 24	3 3 3 24 32	1 1 2 2 1
layer 3	N 4 6 6 32	1 6 6 32 16	1 1 1 1 1
layer 4	N 4 1 1 16	4 1 1 16 2	1 1 1 1 1
layer 5	N 1 1 1 2		
flatten layer	N 2		

Table 2. Architecture of 3-D FCN. 'Valid' padding [21] and 3-D convolution are used in the model, N is the batch size

The architecture of my FCN is presented in table 2. In deep learning, data is divided into many batches having same length which is called batch size, every batch is fed to neural network and neural network minimizes the loss based on batch. N denotes the batch size in table 2, the 3-D FCN is trained to predict if frame 6 of 10 frames is a shot boundary or not, if I change the parameter setting of 3-D FCN, it can also accept input of other length, e.g. input having 20 frames, FCN will predict if frame 6 to 16 are shot boundaries.
At the prediction stage, a snippet can only be predicted as shot boundary or not, I use the central frame to represent the snippet and find the corresponding day of year of that frame, as a result, there are a set of dates, I use the k-means clustering [22] to group these dates into four classes. After discarding the earliest and latest date in each class, the averaged date of rest date is the predicted transition date for each class.

3.3. Siamese Network

In the CNN-based classification, I have spent much time in preprocessing and data augmentation to make the class size balanced, is it possible to use limited unbalanced class to learn the representation of dataset and estimate the transition dates? The answer is using siamese network [23][24][25], aforetioned CNN needs to ensure which class each sample belongs to, however, when number of samples is few or there are too many classes, conventional CNN-based classification does not work well because it can not learn a good representation from few samples. Siamese network learns a similarity metric between during training, and compare or match training samples to test samples based on this learned similarity metric. Siamese network is suitable for scenarios when there are too many classes or too few samples per class. In a siamese network there are two identical sister networks, each taking one of a pair of images, the last layers of two neural networks are fed to contrastive loss function, which is the most special componet of the siamese network, instead of classifying

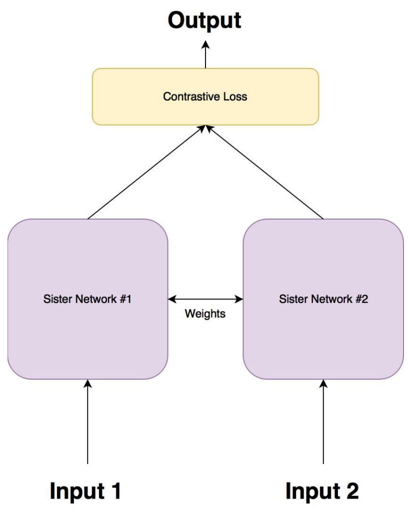

Figure 3. Architecture of siamese network [26], two sister networks share the same weights.

input to exact class, contrastive loss function enables siamese network to measure how different a pair of images are.

$$D = \sqrt{(f(I_1) - f(I_2))^2} \tag{9}$$

$$Loss = \frac{1}{2}yD^2 + \frac{1}{2}(1-y)(max\{0, m-D\})^2 \tag{10}$$

In Equation 9, f is the function performed by sister network, D is the euclidean distance between the two outputs of a pair of inputs. Equation 10 is originally invented by Yann leCun *et al.* in [25], $y = 1$ if a pair of inputs are from same class, otherwise $y = 0$. When $y = 1$, the two inputs are similar, if their euclidean distance is large, then the contrastive loss will also be large; when $y = 0$, $Loss = \frac{1}{2}(max\{0, m-D\})^2$, this is a hinge loss function [27], m is the margin value, if the euclidean distance of outputs from a actually dissimilar pair is beyond this margin, the hinge loss is zero and does not contribute to the contrastive loss, because I only want to optimize the network based on the pairs that are actually dissimilar but the network thinks they are fairly similar.
A CNN plays the role as sister network , which outputs a vector of shape [N 256] where N is the batch size, this vector is fed to contrastive loss function to calculate the similarity of the input pair.
In siamese network there are only the similarity between samples, to predict the transition date, I borrow the idea of image retrieval [28]. Let us call the images from training dataset taken in transition date the transition images hereafter, I feed all test images and transition images to the trained model and get output features of them respectively, then average the output features of transition images to get the transition vector. By calculating the euclidean distance vector between transition vector and output features of test images, I can therefore find the most similar n test images to the transition images, the n is set as 10. After discarding the earliest and latest date of n test images, the averaged date of rest date is the predicted transition date.

3.4. Regression

Besides neural networks mentioned above, I also build a regression network that direcly uses the day of year of the input image as label. The regression network is built by removing the softmax layer from CNN and setting the number of class as one, the rest part of regression network is same with prementioned CNN. The output of regression network is not the class number anymore, but a scalar S, the label I feed to regression network is the corresponding day of year (called D hereafter) of the input image, the regression network is optimized by minimizing the difference between D and S.

$$L_\sigma(a) = \begin{cases} \frac{1}{2}a^2 & \text{if} |a| < \sigma \\ \sigma(|a| - \frac{1}{2}\sigma) & \text{otherwise} \end{cases} \tag{11}$$

Regression problem usually utilizes huber loss function which is defined in equation 11 [29]. σ is a parameter to be set in huber loss, a is difference between output and ground truth, this function is quadratic for small values of a, and linear for large values, compared with least square loss, huber loss lower the penalty for outliers, make our regression network more robust to outliers. The prediction stage of regression network is similar to that of siamese network, each test image get an ouput from trained model in the format of day of year. I find those test images whose outputs are identical or close to transition dates on training dataset, the day of year of those test images are the predicted transition dates of test dataset.

4. Experiments

The objective of my experiments is to find a more accurate algorithm to detect the key phenological transition dates, therefore, I use the human-annotated transition date as baseline, and calculate the gap of days between the transition date detected by algorithms and transition date annotated by human. To evaluate the performance of these algorithms, the smaller the gap is, the better the algorithm is.

4.1. Data used

The images in PhenCam dataset usually not only contain the vegetation but lane or telephone pole, etc., to acquire the region of interest (ROI), images on every site have corresponding mask images, Figure4 and Figure 5 is an example image and corresponding mask image.
Our research data comes from four sites in PheneCam: (1)

Harvard; (2) Bartlettir; (3) Queens; 4) Umichbiological. The images of the above sites have two advantages over images from other sites, first: their images have higher resolution; second, in the images from above four sites, the percentage of forest area of the whole image is higher than those from other sites. These two advantages make it possible to extract more representative information from images. For different methods, the format of input data is different, I will introduce them in experiments settings.

I study the phenology of above sites on year 2008, 2009 and 2010. For site Harvard, Queens and Umichbiological, images are taken every 30 minutes every day in year 2008, 2009 and 2010, images are taken every one hour at site Bartlettir in year 2008, 2009 and 2010, to keep the high quality of images, I only collect images during 10am to 15pm, so there are 10 images every day for site Harvard, Queens and Umichbiological, 5 images for site Bartlettir.

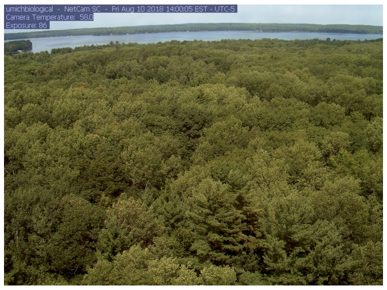
Figure 4. An example image of umichbiological site in 2010, PhenoCam.

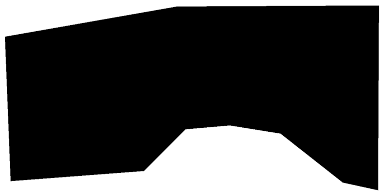
Figure 5. The mask image of site umichbiological, black area is Region of Interest(ROI).

There are 4 events: (1) majority of trees starts leafing out; (2) leaf area reaches maximum; (3) leaves start to change color in fall; (4) majority of trees lost their leaves. I use the label presented by literature [9]. For every event in each year, six human observers look through data in the year and use a common protocal to annotate the day when the event happens in that year as label. When using labels, to reduce inter-observer variability in visually assessed dates, the earliest and latest annotations of each event are discarded.

Data in 2008 and 2009 is used as training data and validation data respectively, data in 2010 is regarded as test data. The models are evaluated on each site data separately.

4.2. Experiments settings

To evaluate the Generalized Sigmoid model, I use the generalized sigmoid function written in matlab[30] by Steve Klosterman to fit the GCC index of year 2010 provided by PhenoCam, the code is available at https://github.com/klostest/PhenoCamAnalysis, the code also contains the function calculating the local extrmas, thus, the returned value is the transition date in format day of year.

The architecture of CNN is shown in Section 3, the input to CNN-based classification model is a batch of RGB images with shape [N 256 256 3], I set N as 64. After the data augmentation, there are about 600 images per class for site Queens, Harvard and Umichbiological respectively, 300 images per class for site Bartlettir. I choose entropy loss function and Adam Optimizer to minimize the loss which are common setting in CNN classification tasks. The condition to stop the training process is when the gap of days between validation data and human-annotated data has reached the minimum.

The input to 3-D FCN is a video snippet, different from other model having image size of [256 256 3], the image size in 3-D FCN is [64 64 3] as 3-D convolution needs more computation, a large image size would make training extremely slow. To cover as much temporal information as possible, I divide the whole dataset into multiple snippets of 10 frames, with an overlap of 5 frames for non-transition frame and overlap of 1 frame for transition frame, there are about 3000 frames for site Queens, Harvard and Umichbiological respectively, 1500 frames for site Bartlettir, thus, there are about 1500 snippets and 40 snippets of them are annotated as transition, for site Queens, Harvard and Umichbiological respectively, 700 snippets for site Bartlettir, and 20 snippets of them are annotated as transition, to make the training data more balanced, I randomly discard half of those non-transition snippets, the rest snippets are fed to 3-D FCN.

The input to siamese network is a pair of images with shape [2*N 256 256 3] where N is batch size. The expanded labels are also used for siamese network which means in training data the former and later 5 days are also regarded as transition date. There is no need to feed images of noise class to siamese network because the network can learn the dissimilarity between four classes representating four events. After excluding the noise class there are about 400 images for site Queens, Umichbiological and Harvard and 200 images totally for site Bartlettir, these images are randomly paired and the label of pair depends on two images have same class or not, I set the number of pairs is 3000, these pairs are fed to siamese network batch by batch, the batch size N is 64.

The setting for regression network is almost same with that of CNN, the difference is there is no sofmax layer and the number of class is set 1, thus the regression network can be trained to predict the day of year of test images.

4.3. Evaluations

Let us denote GS as generalized sigmoid algorithm, CN as CNN-based classification, SN as siamese network and RN as regression network in the following table.

Method	event 1	event 2	event 3	event 4
GS	2	17	8	22
CN	2	16	2	1
SN	3	20	11	9
RN	11	26	15	19
3D FCN	25	39	60	52

Table 3. Harvard Site 2010: Absolute difference to human-recognized label.

Method	event 1	event 2	event 3	event 4
GS	24	7	2	23
CN	5	5	2	4
SN	7	10	4	8
RN	5	14	13	20
3D FCN	45	42	23	70

Table 4. Umichbiological Site 2010: Absolute difference to human-recognized label.

Method	event 1	event 2	event 3	event 4
GS	1	11	8	18
CN	0	8	3	5
SN	4	10	4	8
RN	3	12	17	27
3D FCN	41	52	12	24

Table 5. Queens Site 2010: Absolute difference to human-recognized label.

Method	event 1	event 2	event 3	event 4
GS	1	15	11	23
CN	3	11	6	14
SN	2	9	5	10
RN	10	19	16	24
3D FCN	10	37	66	29

Table 6. Bartlettir Site 2010: Absolute difference to human-recognized label.

Tabel 3, 4, 5 and 6 are the evaulations of four sites. For site harvard, umichbiological and queens, the performance of CNN-based classification is best, it gives the predicted days of event 1 (majority of trees starts leafing out), event 3 (trees first start to change color in the fall), event 4 (majority of trees had lost all leaves) that are close to human recognized, for event 2 (green leaf area is maximum), predicted days of all methods have relatively large gap with human annotated results, I conclude that once the green leaf area reaches maximum the area has a decreasing period and reaches maximum again in summer, which makes model hard to learn this process, I also find that observations of six human observers have relatively large variance in recognizing this event. Siamese network is more accurate than other models in detecting the transition dates on site Bartlettir, this is fair because data of site bartlettir only have half the size of other three sites, performance of siamese network decrease more slightly than that of CNN-based classification model when sample size is reduced. The performance of Regression model is not as accurate as that of classification model and siamese net, the 3-D FCN works poor. I think the poor performance of regression model and 3-D FCN is because temporal information has a strong pattern only during growing seasons (April-June, Sep to Nov). The temporal clues out of vegetation's growing season are weak, which makes it hard for regression and 3-D FCN to learn the temporal pattern, another possible resaon is, in common video dataset, the shot boundaries between frames are highly distinguishable, however, in vegetation dataset, the change of frames is slow and not obvious over time, which means it is hard to capture the temporal pattern.
To summarise the performance of different models on detecting the transition dates of vegetations, CNN-based classification model works best, siamese network and generalized sigmoid model have roughly same performance, they are both good, performance of regression model is not bad but 3-D FCN works poor.

5. Discussion

In this thesis, I introduce few deep learning approaches and how they can be applied in detecting the phenological transition dates, I also compare these deep learning methods with traditional sigmoid-based methods, and find CNN-based classification can yield more accurate results.
There are also many limitations in my work, models exploiting the temporal information is of poor performance, the particularity of vegetation dataset and the inappropriate setting of model are both responsible for the poor result. Actually there is another model which has great potential to achieve success in this task, convolution neural network plus recurrent neural network (RNN), however, it takes much more time to fine-tune the CNN+RNN model, and considering models relying on temporal information have relatively poor performance in experiments, I therefore stop spending time fine tuning the CNN+RNN model, which could be a good approach to perform transition dates

detection in future work. I also have to admit, it usually takes much time to train and fine tune the models to earn better performance.
The scale of dataset in my experiments is not enough to train the best model, the evaluation of performance will become more convicible if data drom more sites and more years is involved in. Integrating data from many sites and many years could be another good way to train the model.

References

[1] Ranga B Myneni, CD Keeling, Compton J Tucker, Ghassem Asrar, and Ramakrishna R Nemani. Increased plant growth in the northern high latitudes from 1981 to 1991. *Nature*, 386(6626):698, 1997.

[2] Michael A White, Peter E Thornton, and Steven W Running. A continental phenology model for monitoring vegetation responses to interannual climatic variability. *Global biogeochemical cycles*, 11(2):217–234, 1997.

[3] Mark D Schwartz. Advancing to full bloom: planning phenological research for the 21st century. *International Journal of Biometeorology*, 42(3):113–118, 1999.

[4] Christian Körner and David Basler. Phenology under global warming. *Science*, 327(5972):1461–1462, 2010.

[5] Xiaoyang Zhang, Mark A Friedl, Crystal B Schaaf, Alan H Strahler, John CF Hodges, Feng Gao, Bradley C Reed, and Alfredo Huete. Monitoring vegetation phenology using modis. *Remote sensing of environment*, 84(3):471–475, 2003.

[6] Oliver Sonnentag, Koen Hufkens, Cory Teshera-Sterne, Adam M Young, Mark Friedl, Bobby H Braswell, Thomas Milliman, John OKeefe, and Andrew D Richardson. Digital repeat photography for phenological research in forest ecosystems. *Agricultural and Forest Meteorology*, 152:159–177, 2012.

[7] Liang Liang, Mark D Schwartz, and Songlin Fei. Validating satellite phenology through intensive ground observation and landscape scaling in a mixed seasonal forest. *Remote Sensing of Environment*, 115(1):143–157, 2011.

[8] Andrew J Elmore, Steven M Guinn, Burke J Minsley, and Andrew D Richardson. Landscape controls on the timing of spring, autumn, and growing season length in mid-atlantic forests. *Global Change Biology*, 18(2):656–674, 2012.

[9] Stephen Klosterman, Koen Hufkens, JM Gray, E Melaas, O Sonnentag, I Lavine, L Mitchell, R Norman, MA Friedl, and Andrew Richardson. Evaluating remote sensing of deciduous forest phenology at multiple spatial scales using phenocam imagery. 2014.

[10] Morris Kline. *Calculus: an intuitive and physical approach*. Courier Corporation, 1998.

[11] Sarah Taghavi Namin, Mohammad Esmaeilzadeh, Mohammad Najafi, Tim B Brown, and Justin O Borevitz. Deep phenotyping: deep learning for temporal phenotype/genotype classification. *Plant methods*, 14(1):66, 2018.

[12] Hulya Yalcin. Plant phenology recognition using deep learning: Deep-pheno. In *Agro-Geoinformatics, 2017 6th International Conference on*, pages 1–5. IEEE, 2017.

[13] Kentaro Kuwata and Ryosuke Shibasaki. Estimating crop yields with deep learning and remotely sensed data. In *Geoscience and Remote Sensing Symposium (IGARSS), 2015 IEEE International*, pages 858–861. IEEE, 2015.

[14] Michael Gygli. Ridiculously fast shot boundary detection with fully convolutional neural networks. *arXiv preprint arXiv:1705.08214*, 2017.

[15] Charles Poynton. *Digital video and HD: Algorithms and Interfaces*. Elsevier, 2012.

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[17] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[19] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[20] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[21] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[22] Kiri Wagstaff, Claire Cardie, Seth Rogers, Stefan Schrödl, et al. Constrained k-means clustering with background knowledge. In *ICML*, volume 1, pages 577–584, 2001.

[23] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a" siamese" time delay neural network. In *Advances in neural information processing systems*, pages 737–744, 1994.

[24] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 539–546. IEEE, 2005.

[25] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *null*, pages 1735–1742. IEEE, 2006.

[26] illustration of siamese network. `https://hackernoon.com`.

[27] Kilian Q Weinberger, John Blitzer, and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances in neural information processing systems*, pages 1473–1480, 2006.

[28] Ritendra Datta, Dhiraj Joshi, Jia Li, and James Z Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys (Csur)*, 40(2):5, 2008.

[29] Peter J Huber et al. Robust estimation of a location parameter. *The annals of mathematical statistics*, 35(1):73–101, 1964.

[30] Users Guide Matlab. The mathworks. *Inc., Natick, MA*, 1992, 1760.

Appendix

Zhaoyang Cheng

August 2018

Contents

1 Label by human

I use the label annotated by Steve Klosterman [1], they evaluate five kinds of transition dates of year data, while we choose four from them since transition date 4 and 5 are too close, which makes it hard for us to expand the label in classification method. The four transition dates in their paper are:

1. when the majority of trees started leafing out
2. when the canopy reached full maturity
3. when the canopy first started to change color in the fall
4. when the majority of trees had lost all leaves.

Following are the transition dates denoted by six observers using a common protocol, in the table we call them event 1 2 3 and 4. NA means not a number, if there is a NA in a row, we discard NA and average the rest of the row as averaged transition date, if not, we remove the biggest and smallest date of the row and averge the rest.

event	year	observer 1	observer 2	observer 3	observer 4	observer 5	observer 6
1	2008	127	128	126	127	NA	129
2	2008	138	148	174	165	NA	155
3	2008	250	241	216	186	252	250
4	2008	305	303	305	303	282	292
1	2009	122	125	121	122	121	126
2	2009	140	146	170	163	140	146
3	2009	250	250	202	218	232	257
4	2009	304	303	305	305	286	305
1	2010	114	117	112	114	115	116
2	2010	125	141	169	180	131	141
3	2010	258	234	202	233	248	256
4	2010	301	301	310	302	290	302

Table 1: Transition dates of Bartlettir site from 2008 to 2010

event	year	observer 1	observer 2	observer 3	observer 4	observer 5	observer 6
1	2008	115	118	117	136	NA	117
2	2008	144	145	166	181	NA	153
3	2008	266	NA	232	239	280	274
4	2008	321	315	353	320	296	317
1	2009	117	121	107	121	120	120
2	2009	136	144	167	182	141	157
3	2009	267	266	251	248	271	271
4	2009	319	314	333	319	298	314
1	2010	113	115	112	113	115	113
2	2010	132	137	156	179	128	146
3	2010	269	NA	NA	259	269	277
4	2010	315	313	329	315	292	314

Table 2: Transition dates of Harvard site from 2008 to 2010

event	year	observer 1	observer 2	observer 3	observer 4	observer 5	observer 6
1	2008	120	120	120	110	110	115
2	2008	NA	NA	163	NA	NA	159
3	2008	266	270	272	259	269	269
4	2008	312	304	349	313	290	305
1	2009	120	122	121	120	122	123
2	2009	135	142	123	179	133	151
3	2009	265	267	253	260	264	268
4	2009	305	303	336	309	284	306
1	2010	105	112	107	109	110	109
2	2010	123	140	150	183	126	136
3	2010	255	263	256	252	261	261
4	2010	300	299	336	313	274	300

Table 3: Transition dates of Queens site from 2008 to 2010

event	year	observer 1	observer 2	observer 3	observer 4	observer 5	observer 6
1	2008	125	126	119	130	135	120
2	2008	150	155	160	156	163	148
3	2008	283	266	NA	NA	NA	282
4	2008	311	309	314	NA	NA	313
1	2009	135	139	110	15	130	NA
2	2009	154	157	161	74	159	NA
3	2009	278	281	271	175	285	NA
4	2009	306	306	319	236	300	NA
1	2010	125	140	93	8	121	119
2	2010	143	157	155	78	144 156]	
3	2010	273	279	256	182	271	271
4	2010	299	306	315	230	284	300

Table 4: Transition dates of Umicnbiological site from 2008 to 2010

2 Deep learning and neural network

Artificial Neural Network (Artificial Neural Network, ANN) is a simulation and approximation of biological neural networks. Adaptive nonlinearity of a large number of neurons connected by dynamic network system. In 1943, psychologist McCulloch and logician Pitts proposed the first mathematical model of neurons-MP model [2]. The MP model is groundbreaking research for later work. In the late 1950s and 60s, Rosenblatt added learning function on the MP model, proposed a single layer perceptron model, it is the first time the neural network is put into practice [3][4]. But the single layer perceptron network model cannot deal with linear indivisible problems. Until 1986, Rumelhart And Hinton *et al.* proposed a training method based on error inverse propagation algorithm. Multi-layer feedforward network -backpropagation network(BP Network) which solved some problem that single layer perceptron cannot solve. Due to the various learning models have been proposed, such as support vectors machine [5], And when adding the number of layers of the neural network, the traditional BP network will encounter problems such as local optimization, over-fitting and gradient diffusion. These have led to the study of deep models being shelved.
In 2006, Hinton *et al.* published a paper [6]. The main points are as follows: 1 The artificial neural network with multiple hidden layers has excellent different feature learning ability; 2 can be "layer-by-layer pre-training" (layer-wise pre-training) to effectively overcome difficulties of deep neural networks in training, which led to research on deep learning, also initiated the another upsurge of artificial neural network. Bengio [7] systematically introduced the network structure and learning algorithm. Currently, CNN [8] is the first candidate to solve images and videos related problem.

3 Convolutional Neural Network

In my scientific paper, different deep learning models are based on the same structure: convolution neural network (CNN), I therefore give many details of CNN in following sections.

3.1 CNN Predecessors Inspired by Neuroscience

In 1962, biologists Hubel and Wiesel found a series of cells with complex structure in visual cortex of cat brain, those cells are sensitive to local region of visual input space which are called receptive field [9]. Receptive field covers the entire visual domain and works locally in the input space, thus receptive field is able to better dig out the strong relationship between natural images. Huber and Wiesel divide these cells into simple type and complex type, simple cells have local receptive fields, and complex cells, which were invariant to shifted or distorted inputs, arranged in a hierarchical fashion. These works provided the early inspiration to later automated vision systems. In 1980, Fukushima proposed a neurocognitive machine(Neocognitron) [10] with a similar structure to the hierarchical model of Huble and Wiesel. Neocognitron consists of simeple layer(S-layer) and complex layer(C-layer), cascaded together in a hierarchical manner, with this architecture, the network proved successful at recognizing simple input patterns irrespective of a shift in the position or considerable distortion in the shape of the input pattern. Significantly, the neocognitron laid the groundwork for the development of CNNs. Later, based on Fukushima's work, LeCun *et al.* use back propogation to train a CNN(LeNet-5) which is the classical CNN architecture, it has good performance in many pattern recognition fields [8].

3.2 Convolution layer

The main difference between fully connected neural network and convolution neural network is that in CNN a hidden layer neuron is only connected to a subset of neurons in the previous layer by convolution operation, this sparse connectivity makes CNN capable to learn features implicitly and save huge computaion.
The basic structure of CNN consists of input layer, convolution layer, pooling layer and fully connected layer. Generally there are many Convolution layers and each of them is followed by a pooling layer. Convolution layer consists of multiple feature maps, each feature map consists of multiple neurons, neurons that lie in the same feature map shares the weight (parameter sharing), thereby reducing the complexity of network by keeping the number of parameters low [11]. The spatial extend of sparse connectivity between the neurons of two convolution layers is called receptive field. The hyperparameters that control the size of the output volume are the depth(number of filters at a layer), stride(for filter movement) and padding(to control spatial size of output). In order to better understand convolution between feature maps, Figure 1 shows the 2-D CNN schematic diagram of convolution operation between feature maps. each of

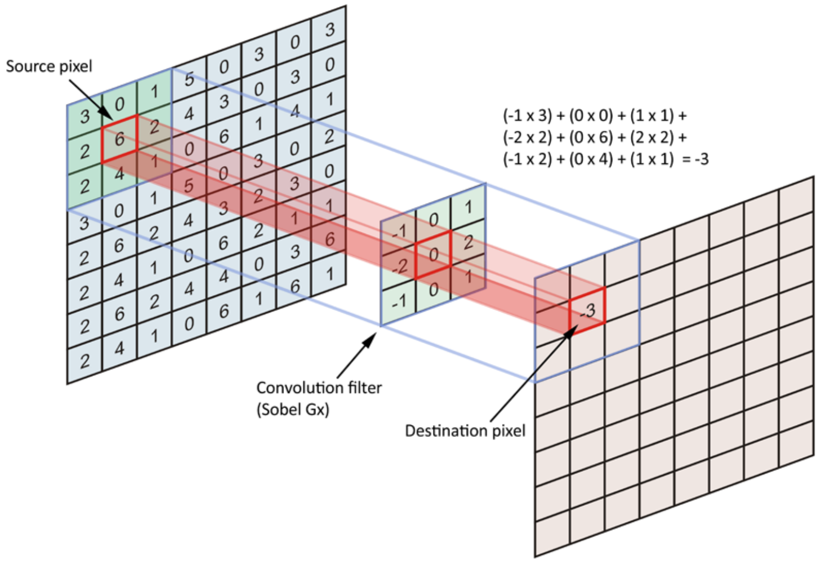
Source pixel
(-1 x 3) + (0 x 0) + (1 x 1) +
(-2 x 2) + (0 x 6) + (2 x 2) +
(-1 x 2) + (0 x 4) + (1 x 1) = -3
Convolution filter
(Sobel Gx)
Destination pixel

Figure 1: example of 2-D convolution

the neurons connect with local region of former feature map through convolution kernels(filters), the convolution kernel is a learnable weighted matrix, the local weighted sum of convolution operation is passed to a non-linear function(activation function),

which results in the corresponding ouput of neurons.
Convolution layer of CNN extracts the different features of the input through convolution operations, the first convolution layer extracts low-level features such as edges, lines, corners, higher-level convolution layer extracts the more advanced features.

3.3 Padding

In convolution operation, the feature map after convolution usually lose the pixels at the border of feature map, padding technique is used to compensate the lose of pixels. There are two kinds of padding, the first one is 'same' padding, 'same' padding pads pixels to the border of images and assign value (usually zero) to those pixels, for the sake of keeping size of image before and after convolution same. Taking a row with length 7 for example, assuming stride is 1, then we add two more pixels to the left and right of this row, yielding another row still with length of 7.

$$height_{new} = ceil(\frac{height_{old}}{stride}) \tag{1}$$

Equation 1 is the calculation of height of image through convolution with 'same' padding. Stride is the step size the convolution kernel moves every time, ceil is function returing the smallest integer that equal or bigger than the input.
Second kind of padding is 'valid' padding, actually, valid padding does not do any padding, convolution stops when kernel meets the pixel that can not be a center pixel of convolution.

$$height_{new} = ceil(\frac{height_{old} - size_{kernel} + 1}{stride}) \tag{2}$$

Equation 2 is the calculation of height of image through convolution with 'valid' padding.

3.4 Activation function

In traditional CNN, the activation function uses a saturating nonlinearity such as sigmoid function, tanh function, etc. Different from saturated nonlinearity function, unsaturated nonlinear function (non-saturating Nonlinearity) can solve the gradient explosion/gradient vanishing problem, it also speeds up the convergence [12]. Jarrett *et al.* [13] explored different rectified nonlinear functions in CNN found that they can significantly improve the performance of CNN, this conclusion was also verified in [14]. In the current CNN structure ,commonly used activation functions in CNN are unsaturated nonlinear functions such as rectified linear unit(ReLU) function. Figure 2 shows the curve of Tanh and RelU function, different from Tanh, the output of ReLU will not saturate as the input increases.

$$f(x) = max(0, x) \tag{3}$$

Chen [15] analyzed 3 factors affecting CNN's performance. Factors: number of layers, number of feature maps, and network architecture. In his report Chinese handwriting recognition experiments use CNN with 9 structures, some conclusions on test results from CNN with a relatively small convolution kernel : 1) increasing the depth of the

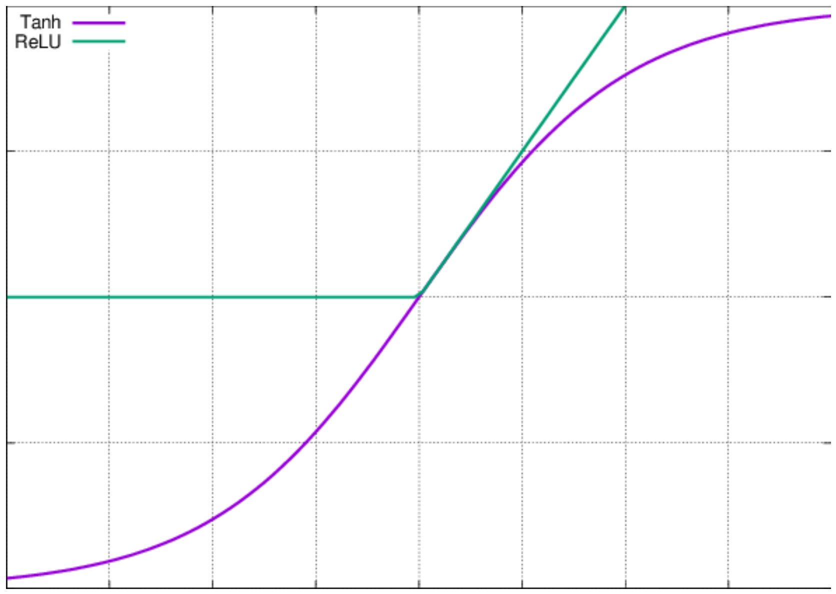

Figure 2: ReLU function and Tanh function

network can improve accuracy Rate; 2) increasing the number of feature maps can also improve accuracy; 3) Adding a convolution layer can increase higher accuracy by adding a fully connected layer. Literatur [16] pointed out that the deep network structure has two Advantages: 1) can promote the reuse of features; 2) can obtain more abstract features in high-level representaions, due to that highly abstract concepts can be constructed based on the lower abstract concepts, so the deep structure can get more abstract representations, such as pooling operations in CNN can create this abstraction, highly abstract concept is invariant to the lost local change of inputs.

3.5 Pooling layer

The pooling layer follows after the convolution layer, it also consists of multiple feature maps, each of its feature map uniquely corresponds to a feature map of the previous convolution layer. Poolint layer is designed to retain space invariant features by reducing the resolution of the feature maps. The pooling layer acts as a secondary extraction of features, every neuron of pooling layer conducts pooling operation on local region of corresponding previous convolution layer The commonly used pooling method is max pooling, taking the largest value in local region, mean pooling, averaging all values in the local region, stachastic pooling [17][18]. Literature [19] gives detailed theoretical analysis of max pooling and mean pooling, there are some conclusions: 1) max pooling is especially suitable for separate very sparse features; 2) using all points in the local area to perform pooling operations may not be optimal, such as averaging pooling, using all points in the local region.

3.6 Fully connected layer

In the CNN, one or more fully connected layers(FC layer) are added after multiple convolution layers and pooling layers. Each neuron in the FC layer is fully connected with all the neurons in the previous layer . Fully connected layer can integrate category-specific local information in convolution layer or pooling [20]. In order to improve CNN Network performance, the activation function of each neuron in the fully connected layer is usually the ReLU function.
The output value of the last layer of the fully connected layer is passed to the output layer which is also called softmax layer, bacause output layer uses softmax logistic regression for classification. For a specific classification Task, it is very important to choose a suitable loss function. Literature [21] introduced several commonly used loss functions of CNN and analyzed their respective characteristics.
When a large feedforward neural network is trained on a small amount of dataset, due to high capacity of neural network, it usually does not perform well on the held-out test data. To avoid overfitting, commonly used regularization method in the fully connected layer is dropout or dropconnect technique,

$$f(x) = max(0, x) \tag{4}$$

In the Figure 3, (i) shows a standard neural network. Applying Dropout is shown in (ii)

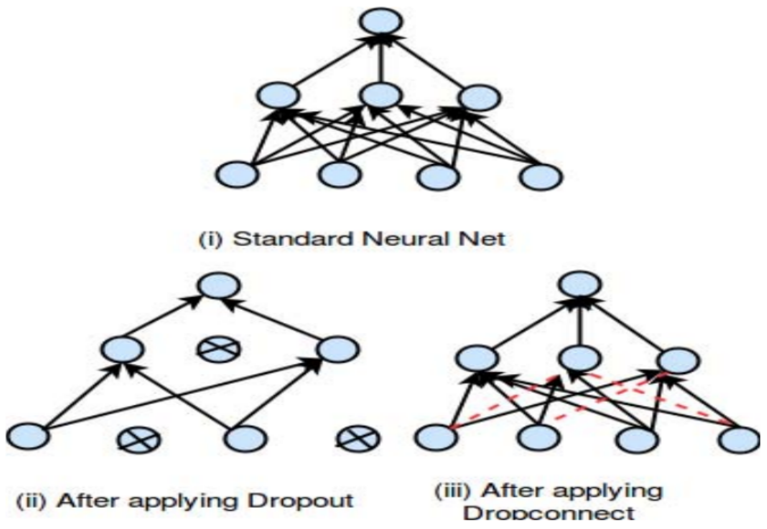

Figure 3: Dropout and DropConnect[22]

is equivalent to sampling a neural network. The activations of some of the neurons are set to zero during forward and backward prapogation of training. While DropConnect randomly set the link weights to zero, this is marked red in (iii) in the Figure.
Due to the randomness of dropout technique, its corresponding network structure is not the same, but all of these structures share weights, one Neuron's existence does not depend on other specific neurons, so this technique reduce the adaptive complexity of inter-neuronal and make network learns more robust features. Currently, for CNN,

most of the research uses ReLU+dropout technique and has achieved very good results [23][24][25].

3.7 Feature map

The number of feature maps is an important parameter of CNN, usually it is set according to the actual application, if the number of feature maps is too small, some features that are conducive to network learning may be ignored; but if the number of feature maps is too large, the number of training parameters and network training time will also increase. Literature [26] proposed a theoretical method used to determine the optimal number of feature maps. The literature found, compared to the CNN structure having same number of feature maps per layer , the pyramid architecture (the number of feature maps of this structure is increased by a factor of multiple) can exploit computation resources more efficiently. At present, the setting of the number of feature maps based on the experiment and observation.

4 Literature review on video classification

As the images and videos are increasing on the Internet with unprecedented speed, efficient videos classification algorithm helping people find satisfactory content is in demand. Video classification algorithms are supposed to understand the content of videos or video clips and classify them automatically to one or more labels.
The huge volume of video data has motivated approaches to automatically categorizing video contents according to prominent classes such as human activities. There is a large body of literature focusing on computing effective local feature descriptors (e.g., HoG [27], HoF, etc.) from spatio-temporal volumes to account for temporal clues in videos. These features are then quantized into bag-of-words which are further fed into classifiers like support vector machine (SVM) [28]. In contrast with hand-crafted features which is time-consuming and requires domain knowledge, the representation of features extracted by deep neural network becomes prevalent. There are two categories of deep learning models for video classification: convolution neural network based video classification and convolution neural network plus recurrent neural network (RNN) video classification.
Why RNN palys an important role in video classification? Let me explain what is RNN first.

4.1 Recurrent neural network

Recurrent neural network is an important class of artificial neural network, which has achieved great success on many tasks in computer vision (CV) and natural language processing (NLP). This review will summarize the application of recurrent neural network on video classification.
Let us start with the difference between feed forward network and recurrent neural network. In feed forward neural network, inputs are fed to network and transformed to an

output, the output at this moment does not depend on the outputs of previous inputs, for example, in handwritten digits classification task, that the first digit is predicted as one will not has influence on the 100th digit being predicted as three. But the situation is different in recurrent neural network, the decision of recurrent neural network at this moment depends on the decision the network made at previous moments, as a result, the output of this moment is a weighted combination of previous output and current input, which also means recurrent neural network has a memory capturing what has been calculated so far, in theory recurrent neural network can take input of arbitrarily long, in practice we limit the network only to look at few previous moments. Figure

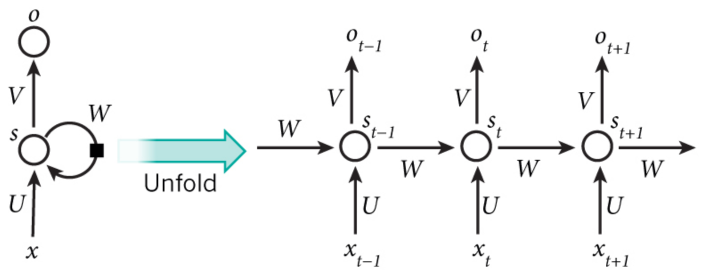

Figure 4: A recurrent neural network and the unfolding in time of the computation involved in its forward computation [29]

4 shows a recurrent neural network being unfolded into a full network which has the same length as the sequence, x_t is the input at time step t, usually an one-hot vector corresponding to an element in a sequence, s_t is the hidden state at time step t, which is calculated based in the previous hidden state and current input:

$$s_t = f(U * x_t + W * s_{t-1}) \tag{5}$$

f is activation function such as tanh or ReLu [14], o_t is output at time step t:

$$o_t = V * s_t \tag{6}$$

Unlike feed forward neural network which has different parameters in every layer, recurrent neural network share parameters (U, V, W in the diagram), across all time steps, which proves RNN is doing same task on different element in a sequence.
I have introduced the basic idea of recurrent neural network, the most popular version of recurrent neural network is Long Short Term Memory recurrent neural network (LSTM) [30].

4.2 Long short term memory network

When handling time-series data, sometimes we only take recent information into consideration, however, there are many occasions where we need to look for further information, the problem of standard recurrent neural network is that it is very hard for it to keep track of information of many steps ago, why? intuitively information declines through propagation, theoretically there is a vanishing gradients problem.

4.2.1 Vanishing gradients problem

The vanishing gradients problem is originally discovered by Sepp Hochreiter [31] and analyzed by Yoshua Bengio [32][33]. when training a feed froward neural network, the error is calculated on the output layer and propagated back to hidden layers, then the weights between units are updated based on the gradients of error, such that the error is minimized, we usually call this process Back Propagation. In recurrent neural network, propagation has two directions, through layers and through time, for example in Figure 4, the error term of individual neuron at layer l and time t δ_t^l flows in two directions, one is to last layer resulting δ_t^{l-1},determined by weight U ,another direction is along time series to time t_1, resulting δ_1^l , determined by weight W. {U, V, W}, is weight of input-to-hidden,hidden-to-output,hidden-to-hidden respectively.
For notation, $o_t = Vs_t$, $\hat{y}_t$ is input to other layers at time t, f is an activation function.

$$\hat{y}_t \quad = \text{softmax}(Vs_t) \tag{7}$$

we will focus on the back propagation through time and leave aside the back propagation through layers which is similar to that in feed forward neural network.
At time t, for one neuron, error is $e_t = -y_t log(\hat{y}_t)$,then sum up errors of all time step, define our loss as :

$$E = \sum_t -y_t log(\hat{y}_t) \tag{8}$$

log function is monotonically increasing. In back propagation through layers, we calculate gradient of error with respect to weight V(hidden-to-output), now in this through time version, we want to calculate gradient of error with respect to weight W($hidden_{t-1} - to - hidden_t$), similar to the way errors are cumulated in a time series, we also sum up gradients of error along time series, which is

$$\partial E/\partial W = \sum_t \partial e_t/\partial W \tag{9}$$

with chain rule, we compute e_t first, that is

$$\partial e_t/\partial W = \frac{\partial e_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial S_t}\frac{\partial S_t}{\partial W} \tag{10}$$

here, $\hat{y}_t = softmax(VS_t)$,$S_t = tanh(WS_{t-1} + Ux_t)$, however, we can not treat S_{t-1} as a constant, have to get more time information.

$$\partial e_t/\partial W = \sum_{k=0}^{t} \frac{\partial e_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial S_t}\frac{\partial S_t}{\partial S_k}\frac{\partial S_k}{\partial W} \tag{11}$$

To investigate the vanishing gradient problem, let's Look at this equation, we should make $\frac{\partial S_t}{\partial S_k}$ more fundemental, $\frac{\partial S_t}{\partial S_{t-1}} \cdots \frac{\partial S_{k+1}}{\partial S_k}$, $\frac{\partial S_t}{\partial S_k} = \prod_{j=k+1}^{t} \frac{\partial S_j}{S_{j-1}}$

$$\partial e_t/\partial W = \textstyle\sum_{k=0}^{t} \frac{\partial e_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial S_t}(\displaystyle\prod_{j=k+1}^{t} \textstyle\frac{\partial S_j}{\partial S_{j-1}})\frac{\partial S_k}{\partial W} \tag{12}$$

$$\frac{\partial S_j}{\partial S_{j-1}} = \frac{\partial S_j}{\partial net_j}\frac{\partial net_j}{\partial s_{j-1}} = f'(net_j) \cdot W \tag{13}$$

where net_t is a weighted input from a hidden layer t-1 to another hidden layer t, $f'(net_j)$ is a jacobian matrix, which means

$$f'(net_j) = diag[f'(net_j)] \tag{14}$$

Let us denote $\eta = |f'(net_j)| \cdot |W|$ Bengio *et al.* [33] has proven that $\eta < 1$, which means by a series of multiplication the gradient will approach zero.

4.2.2 Architecture of LSTM

Having figured out why gradients vanish, researchers find that the solution lies in how to control the long dependencies of information, in LSTM, they use gating mechanism to protect and control the dependencies which is called cell state c_t at time t in LSTM. In Figure 5, LSTM network at time step t has three input parts: input x_t of net, output h_{t-1} from net at last time , cell state c_{t-1} at last time , and two output parts: output h_t from net at time t and current cell state c_t.

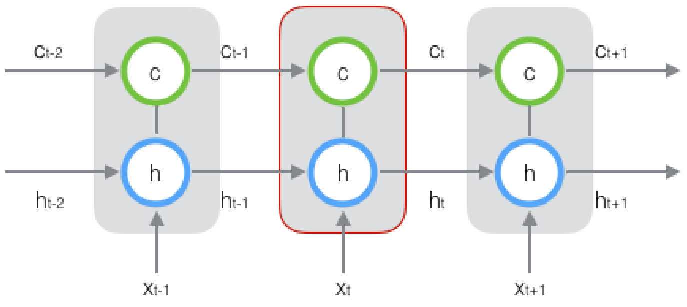

Figure 5: cell state [34]

First they use **forget gate** to control how much information of last cell state c_{t-1} is remained in current cell state c_t.

$$f_t = \sigma(\begin{bmatrix} W_f \end{bmatrix} \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix}) = \sigma(\begin{bmatrix} W_{fh} & W_{fx} \end{bmatrix} \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix}) = \sigma(W_{fh}h_{t-1} + W_{fx}x_t) \tag{15}$$

Then **input gate** is used to control how much information of x_t and h_{t-1} is stored in c_t,

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) = \sigma(W_{ih} \cdot h_{t-1} + W_{ic} \cdot c_t + b_i) \tag{16}$$

Let us use $\tilde{c}_t$ denotes the memory candidate waiting to be added to c_t,

$$\tilde{c}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{17}$$

When information in last cell state c_{t-1} and candidate cell state $\tilde{c}_t$ pass their gate, forget gate f_t and input gate i_t respectively, part of their information is abandoned and the rest is maintained, then they are passed to update cell state c_t.

$$c_t = f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t \tag{18}$$

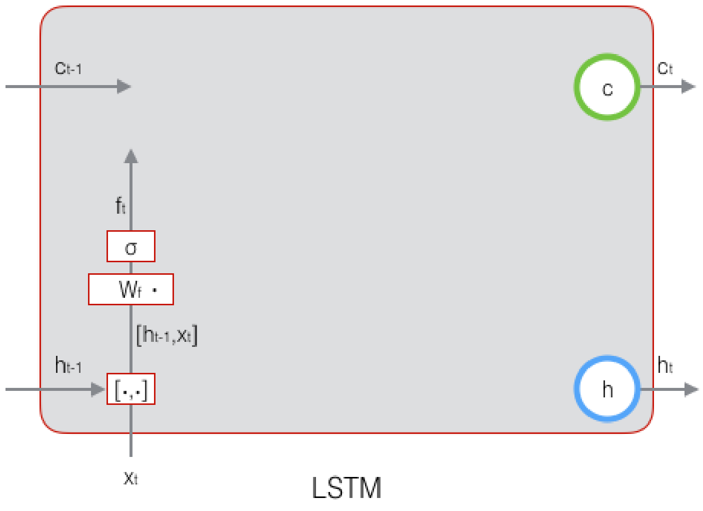

Figure 6: cell state [34]

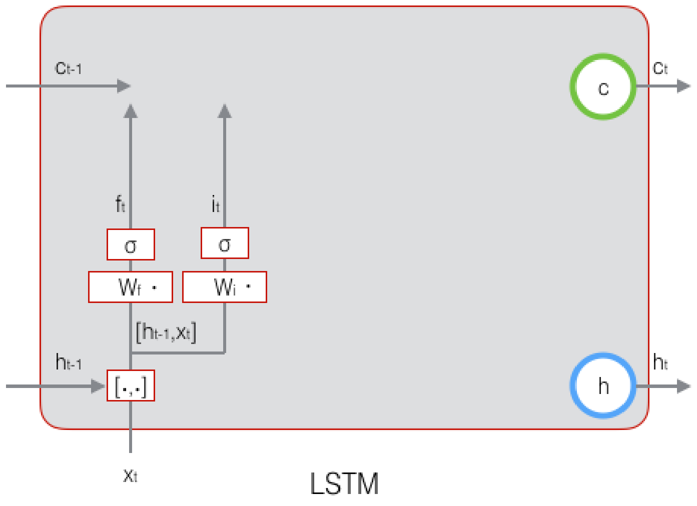

Figure 7: cell state [34]

The **output gate** determines the output of LSTM.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{19}$$

$$h_t = o_t \cdot tanh(c_t) \tag{20}$$

So far, the basic structure of LSTM has been introduced, unlike the traditional recurrent unit which overwrites its content at each time-step, a LSTM unit is able to decide whether to keep the existing memory via the introduced gates. Intuitively, if the LSTM unit detects an important feature from an input sequence at early stage, it easily carries this information (the existence of the feature) over a long distance, hence, capturing potential long-distance dependencies [35].

Having introduced RNN, I will introduce the CNN-based video classification and CNN-RNN based video classification.

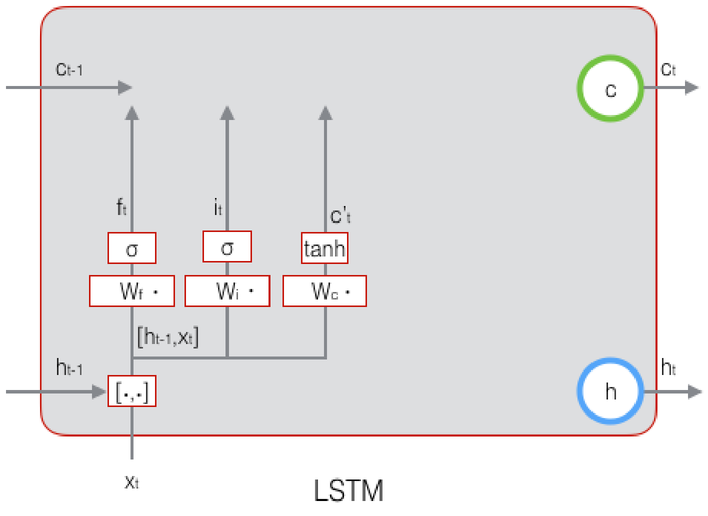

Figure 8: cell state [34]

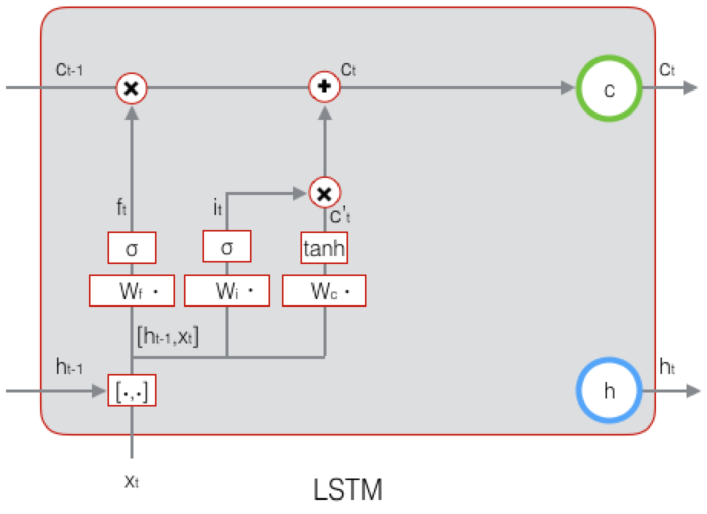

Figure 9: cell state [34]

4.3 CNN-based video classification

The success of CNN on image classification has simulated the utilization of deep features for video classification. The general idea is to treat video clips as a set of frames, for each frame, feature representation is extracted from an advanced CNN, e.g. AlexNet [23], VGGNet [36] or GoogleNet [37]. Frame-level features are averaged into video-level representations as input of classifier such as SVM. More recently, Karpathy *et al.*[38] extended the CNN to work on the temporal dimension by stacking frames over time, Simonyan *et al.*[39] propose a two-stream CNN approach, which uses two CNNs on static frames and optical flows respectively to capture the spatial and the motion information.
These methods, however, only consider static or short-term clues in videos, which is not sufficient for video classification, as many complex contents can be better identified

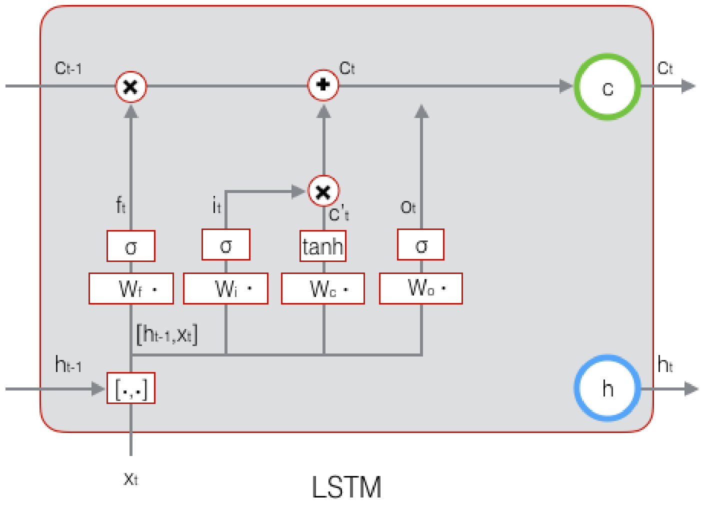

Figure 10: cell state [34]

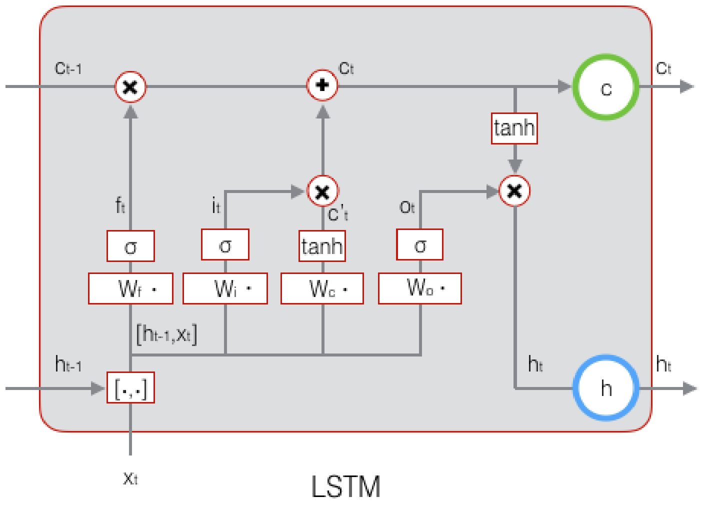

Figure 11: cell state [34]

by considering the long-term clues.

4.4 CNN-RNN based video classification

To address the aforementioned limitations, Wu *et al.* [40] combine CNN and RNN (LSTM) which leverages more temporal information as a hybrid network for video classification. many approaches fuse multiple features in a very “naive” manner by either concatenating the features before classification or averaging the predictions of classifiers trained using different features separately. In this work they integrate the spatial and the short-term motion features in a deep neural network with carefully designed regularizations to explore feature correlations. This hybrid network can perform video classification within the same network and further combining its outputs with the predictions of the LSTM can lead to very competitive classification performance.

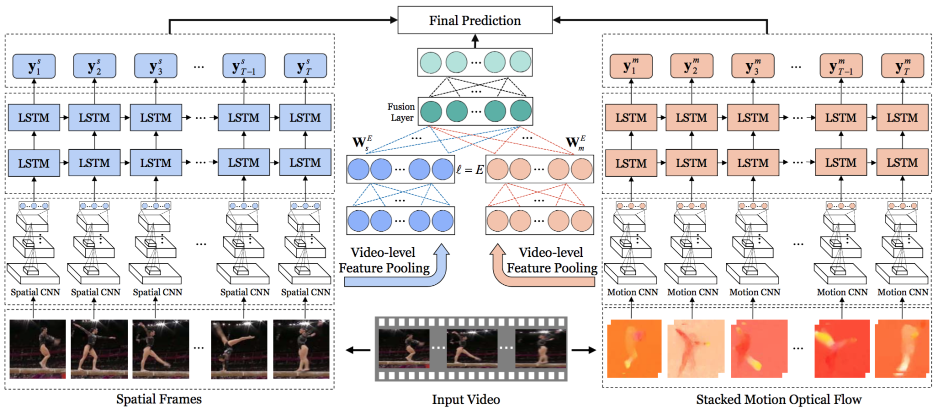

Figure 12: An overview of the proposed hybrid deep learning framework [40] for video classification. Given an input video, two types of features are extracted using the CNN from spatial frames and short-term stacked motion optical flows respectively. The features are separately fed into two sets of LSTM networks for long-term temporal modeling (Left and Right). In addition, they also employ a regularized feature fusion network to perform video-level feature fusion and classification (Middle). The outputs of the sequence-based LSTM and the video-level feature fusion network are combined to generate the final prediction.

In Experiments, they use two popular datasets:
UCF-101 dataset[41], UCF-101 dataset is a action recognition benchmark, which consists of 13,320 video clips of 101 human actions (27 hours in total). The 101 classes are divided into five groups: Body-Motion, Human-Human Interactions, Human-Object Interactions, Playing Musical Instruments and Sports.
Columbia Consumer videos(CCV) dataset[42],The CCV dataset consists of 9,317 YouTube videos annotated with 20 classes, which are mainly events like “basketball”, “birthday party” and “parade”.
Their first experiment proves the effectiveness of LSTM. The second experiment compare their hybrid network with conventional "features+classifier" pattern, which shows with or without the regularized fusion network, the CNN+LSTM network will always perform better then aforementioned method. The last experiment aims at comparing their model to those state-of-the-art models, which also ends with hybrid network having highest accuracies on UCF-101 and CCV dataset.

References

[1] Stephen Klosterman, Koen Hufkens, JM Gray, E Melaas, O Sonnentag, I Lavine, L Mitchell, R Norman, MA Friedl, and Andrew Richardson. Evaluating remote sensing of deciduous forest phenology at multiple spatial scales using phenocam imagery. 2014.

[2] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[3] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[4] Frank Rosenblatt. Principles of neurodynamics. 1962.

[5] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[6] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[7] Yoshua Bengio et al. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[8] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[9] David H Hubel and Torsten N Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106–154, 1962.

[10] Kunihiko Fukushima. Neocognitron–a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *NHK*, (15):p106–115, 1981.

[11] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[12] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

[13] Kevin Jarrett, Koray Kavukcuoglu, Yann LeCun, et al. What is the best multi-stage architecture for object recognition? In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2146–2153. IEEE, 2009.

[14] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[15] Xu Chen. Convolution neural networks for chinese handwriting recognition.

[16] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[17] Y-Lan Boureau, Nicolas Le Roux, Francis Bach, Jean Ponce, and Yann LeCun. Ask the locals: multi-way local pooling for image recognition. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2651–2658. IEEE, 2011.

[18] Matthew D Zeiler and Rob Fergus. Stochastic pooling for regularization of deep convolutional neural networks. *arXiv preprint arXiv:1301.3557*, 2013.

[19] Y-Lan Boureau, Jean Ponce, and Yann LeCun. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 111–118, 2010.

[20] Tara N Sainath, Abdel-rahman Mohamed, Brian Kingsbury, and Bhuvana Ramabhadran. Deep convolutional neural networks for lvcsr. In *Acoustics, speech and signal processing (ICASSP), 2013 IEEE international conference on*, pages 8614–8618. IEEE, 2013.

[21] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Li Wang, Gang Wang, et al. Recent advances in convolutional neural networks. *arXiv preprint arXiv:1512.07108*, 2015.

[22] Neena Aloysius and M Geetha. A review on deep convolutional neural networks. In *Communication and Signal Processing (ICCSP), 2017 International Conference on*, pages 0588–0592. IEEE, 2017.

[23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[24] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[25] Tara N Sainath, Brian Kingsbury, George Saon, Hagen Soltau, Abdel-rahman Mohamed, George Dahl, and Bhuvana Ramabhadran. Deep convolutional neural networks for large-scale speech tasks. *Neural Networks*, 64:39–48, 2015.

[26] Joseph Lin Chu and Adam Krzyżak. Analysis of feature maps selection in supervised learning using convolutional neural networks. In *Canadian Conference on Artificial Intelligence*, pages 59–70. Springer, 2014.

[27] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[28] Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28, 1998.

[29] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[30] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. 1999.

[31] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

[32] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[33] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318, 2013.

[34] colah. understanding lstm, 2017.

[35] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[36] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[37] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, et al. Going deeper with convolutions. Cvpr, 2015.

[38] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[39] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, pages 568–576, 2014.

[40] Zuxuan Wu, Xi Wang, Yu-Gang Jiang, Hao Ye, and Xiangyang Xue. Modeling spatial-temporal clues in a hybrid deep learning framework for video classification. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 461–470. ACM, 2015.

[41] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[42] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 2, pages 416–423. IEEE, 2001.