Towards AI Conversing : FloodBot using Deep Learning Model Stacks

1st Bipendra Basnyat
UMBC
b125@umbc.edu

2nd Nirmalya Roy
UMBC
nroy@umbc.edu

3th Aryya Gangopadhyay
UMBC
gangopad@umbc.edu

***Abstract*—Talking to the electronic device and getting the required information at a minimal time has become today's norm. Although AI-powered conversational agents have percolated the commercial market, their use in a communal setting is still evolving. We postulate that the deployments of chatbots in disaster-prone areas can be beneficial to watch, monitor, and warn people during the crisis. Furthermore, the successful implementation of such technology can be life-saving.**

In this work, we discuss our deployment of a real-time flood monitoring chatbot called FloodBot. We collect, annotate and visually parse images from potentially hazardous areas. We detect the flood conditions and identify objects in harm's way by stacking deep learning models such as a convolutional neural network (CNN), single-shot multi-box object detection (SSD). We then feed the image contents to a knowledge base of our artificially intelligent FloodBot and explore its AI-Conversing power using end to end memory network. We also showcase the power of cross-domain transfer learning and model fusion techniques.

***Index Terms*—Chatbot, Deep Learning, Computer Vision, Deep Natural Language Processing, Mobile Computing**

I. Introduction

Every time there is a disaster, people scroll through various news sources, social media, and television to get an update on the current/on-going event. The information thus gained may be scattered, often irrelevant, or diluted. Media curates information far away from the epicenter of the event and such content can either be irrelevant or less contextual to the actual stakeholders, and disaster struck local inhabitants. A quick status update from the local representative is more valuable than trying to assimilate information from heterogeneous sources.

We argue that an artificially intelligent chatbot could be one such local agent. While Alexa, Google Assistant, Siri, Cortana, and Bixby have weaved their presence into our lives, they can only answer queries about general things. There is still a scarcity of the AI-powered chatbots, mainly designed to disseminate information about the disaster at the grass-root level. We postulate that people will want to know about the potential danger in the area by asking questions. They would rather get summarized and relevant information instead of reading unnecessary contents or waiting for social media posts.

Interest in machine conversation can be traced back to the 1950s when researchers were exploring machine's abilities to think with "The Turing Test". Though some exciting works have happened in the past, it is now that the area of intelligent chatbots and dialog systems is highly energized. The recent rise in interest around chatbots can be attributed to the growing sophistication and accuracy of artificial intelligence and machine learning capabilities. More specifically, deep learning in the natural language process, computer vision, and machine's ability to understand the natural language and high availability of computing power have revived current research paradigm. Developing intelligent chatbots and dialogue systems have been a topic of interest for both the commercial and academic areas. For the industry, it is the cost-cutting measure, while for academia, this is a profound research problem.

For industries, chatbots can automate customer services, reduce human labor costs and streamline their work order. Most of the call centers have either eliminated consumer support persons or significantly streamlined their workforce. Such a reduction in the workforce can be attributed to the successful implementation of chatbots. Chatbots can easily handle most of the mundane tasks. Even for complex situations, the chatbot can pre-gather the required information before transferring the call to a real person and thereby increasing the effectiveness of the problem resolution.

On the other hand, the academic interest in chatbot development is fueled by the complexity of the problem domain and its intricacy with various state of the art machine learning algorithms. The chatbot may appear as a simple question and answering agent, nonetheless several complex algorithms and logical thinking make bot a reality. Building chatbot requires many state of the art machine learning models from natural language understanding to processing complex human generated data. Chatbot development also requires the machine to understand the underlying semantics of user input utterance and generating coherent and meaningful responses.

Another prominent use of chatbots is in the news media. Today, people consume information via social media such as Twitter, Facebook, and WhatsApp. We are getting accustomed to talking to our smart speakers and asking relevant questions. As we get more comfortable in conversing with such agents, the demand for chatbots is poised to rise. Likewise, people will seek information about the potential local hazard in the area by asking questions. They would rather get short real-time and relevant information instead of reading news or waiting for social media posts. Chatbots will need contextual intelligence and domain-specific knowledge to answer the queries about a potential hazard in the local context. Therefore, in this

978-1-7281-6997-2/20/$31.00 ©2020 IEEE
DOI 10.1109/SMARTCOMP50058.2020.00025

Authorized licensed use limited to: University of Maryland Baltimore Cty. Downloaded on November 07,2020 at 23:59:00 UTC from IEEE Xplore. Restrictions apply.

paper, we devise one such domain-specific chatbot. We call it a **FloodBot** and outline steps involved in crafting it. We then describe our first-hand experience from the real-world deployment of FloodBot. -

Key Contributions

The main contributions of this paper are:

- *Cross-Domain Transfer Learning* : We propose a cross-domain transfer learning in a rather uncharted domain of flood detection. We use pre-trained deep learning vision models and fine-tune them to infer ongoing flood potential and severity.
- *Deep Learning Model Stacking*: We propose three different kinds of deep learning models, and stack them to enable AI Conversing power of the FloodBot.
- *Rule Based Bulk Image Annotation*: Data annotation is a resource-intensive task, especially on a larger data set like ours (16K image frames). Thus, in this paper, we propose a noble software-based image annotation technique to annotate images in bulk.
- *Domain Specific Q & A Corpus Enrichment*: Starting with an open-source Q & A corpus as a seed bank, we infuse flood-related Q & A datasets to enable cohesive conversation between FloodBot and human.

In Section II, we present some state-of-the-art work that inspired us. In Section III we discuss our overall Architecture. Section IV highlights our use of cross domain transfer learning techniques. In section V we talk about the end to end memory learning model. We then discuss our data collection in Section VI. Section VII outlines our experiment and evaluations. Section VIII provides the results of the implementation. We share our insights Section IX and conclude in Section X .

II. Related Work

The conversational AI research community defines chatbots as an intelligent agent that can make engaging conversations with a human counterpart. FloodBot's functioning relies on recognizing the scene context and conversing about it.

A. Types of Chatbot

Chatbot or conversational agents take natural language uttered by a user as input and respond. There are two main ways to generate responses. The traditional approach uses hard-coded templates and rules. The more novel approach is to use neural network-based deep learning frameworks. This kind of generative model is trained on large dialogue corpus. Chatbots then learn to generate relevant and grammatically correct responses to the user input. There are two main types of chatbots. The classic type of Chatbot is *Goal Oriented Dialogue System*. This kind of Chatbot posses limited conversational capabilities. However, it is very robust at executing task-specific commands. Task-oriented Chatbots are built to accomplish specific tasks like making restaurant reservations[4, 6]. The other groundbreaking work in the goal-oriented dialogue system was proposed by [15]. The second and more challenging type of Chatbot is *Open-Domain Dialogue System*.These are dialogue agents that are developed for general purposes and are expected to converse or imitate human dialogue capability. They are usually trained with large human conversation databases. The training data volume must be enormous for these types of dialogue systems. Therefore, many successful benchmark works are based on publicly available datasets like movie scripts, twitter data or web scraped question/answer pairs [16, 14, 13]. For these models, there is no well-defined goal. However, they are required to have a certain amount of world knowledge and commonsense to have a logical conversation. The success of a fully functional Open Domain dialogue system has been an active research area and is still far from perfection. In the meantime, a hybrid approach [19] is also being explored. Hybrid chatbots are developed with some level of non-contextual conversing capability. The proposed FloodBot belongs to the hybrid category where we have used large corpus from the different domain and infused flood data.

B. Computer Vision and Natural Language Processing

FloodBot is designed to converse and inform about the situation remotely. Thus the overarching research area comes under Visual Question and Answering [1]. There are many exciting works in the area of deep learning, specifically in the area of Visual Question and Answering (VQA). VQA has the potential to build an end to end deep learning models for real-life cases. Multi-modal image captioning tries to develop a joint (image + text) model [7] to interpret the content in an image. Deep learning and vision model can provide a description which not only identifies the objects but also infers activities they are involved in. [17] For a successful chatbot implementation, the output needs to be coherent in answering the multiple incoming questions. A solution for this has been proposed by [8] as the Dynamic Memory Network (DMN). DMN is a neural network-based framework for general question-answer tasks that are trained using raw input-question-answer triplets. [19] is another state of the art architecture in solving sequence tagging tasks, classification problems, sequence-to-sequence tasks, and question answering tasks that require transitive reasoning.

The significant challenges in all the areas discussed above have been the ability to create one unified model. There are some generalized solutions, but they are mostly task-oriented. Due to the complexity of the problem, researchers are only focusing on individual area. This hinders the use of deep learning in delivering end to end problem solution. Therefore, in this work, we try to implement these technologies together and propose a joint multi-modal learning technique to solve real world problem.

III. Overall Architecture

The objective of this work is to observe the current situation of potential flood areas, infer other contextual information, and converse about potential risk. In order to achieve this goal, we propose three deep learning models: two vision-based and one language-based. Vision models understand the scene, and the language model empowers the FloodBot to meet linguistic requirements.

Authorized licensed use limited to: University of Maryland Baltimore Cty. Downloaded on November 07,2020 at 23:59:00 UTC from IEEE Xplore. Restrictions apply.

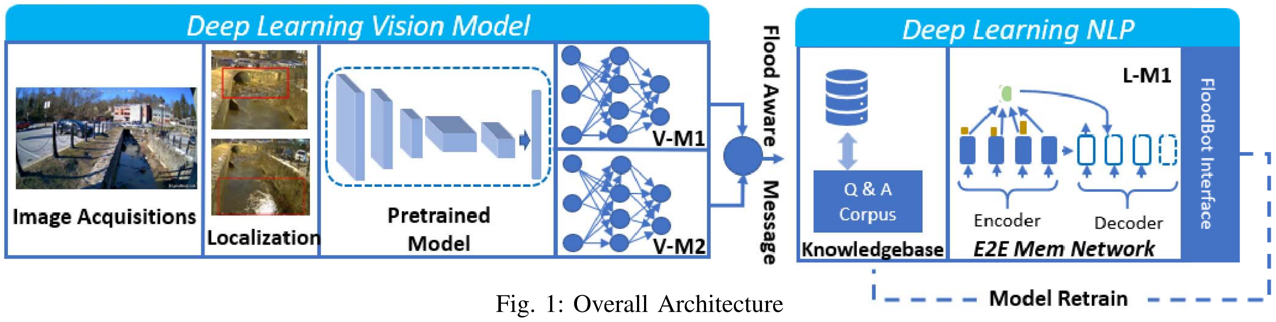
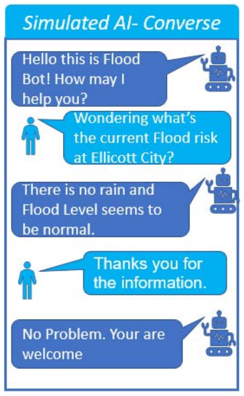


Fig. 1: Overall Architecture

Two parallel deep learning vision models observe the flooding condition and scene. The first one is a Flood detection model called **V-M1 (Vision Model1)**. The second model is a Single-shot Multi-box object detection model called **V-M2 (Vision Model2)**. The joint learning from these two models are propagated to the third and final model called **L-M1 (Language Model1)**. Inferences from these three models provide the required knowledge-base for AI-Conversing FloodBot. The endpoint of our implementation is an *AI conversing*-chatbot called **FloodBot**. FloodBot is a user interface for computers and humans to communicate. The deployment of our system follows a native client-server architecture. The clients are cameras, and other sensing nodes deployed on the ground while servers side include GPU powered deep learning server. We show a simulated AI-conversation in Figure 2 between FloodBot and human. The FloodBot is expected to understand the intent and utterance of the human counterpart and provide information as requested via one of the user inputs (typing in a computer interface). Once the model is trained to converse via a keystroke, it can be easily converted into an automatic speech recognition system.



Simulated AI- Converse
Hello this is Flood Bot! How may I help you?
Wondering what's the current Flood risk at Ellicott City?
There is no rain and Flood Level seems to be normal.
Thanks you for the information.
No Problem. Your are welcome

Fig. 2: AI-Converse

In this work, we use two state of the art deep learning frameworks: Cross Domain Transfer Learning and End to End Memory Network. They are the binding principle for our Floodbot. In the following sections we briefly review them and present our implementation and their relevance to our work.

IV. Cross Domain Transfer Learning

Traditional supervised machine learning methods typically require sufficient labelled training instances to construct accurate models. In practice, however, only limited labelled training instances are available. Transfer learning is the improvement of learning in a new task through the transfer of knowledge from a related task that has already been learned [20].

In transfer learning we deal with the data $\mathcal{D}$ coming from *Source* or *Target* Domains and the probability distribution $\mathcal{P}$ there to. The source domain is expected to be knowledgeable and containing ample annotated data. The main goal of Transfer learning is to find an optimized learning function $f(\cdot)$ which maps $\mathcal{Y}_s$ and $\mathcal{Y}_t$ (class labels from source and target domain respectively). We can formally define transfer learning as: Given a labelled source domain:

$$\mathrm{D}_s = \{\mathbf{x}_i, y_i\}_{i=1}^{n} and \mathcal{D}_t = \{\mathbf{x}_j\}_{j=n+1}^{n+m}; \mathcal{P}(\mathbf{x}_s) \neq \mathcal{P}(\mathbf{x}_t)$$

The purpose of transfer learning is to use the knowledge of $\mathcal{D}_s$ and transfer the learned knowledge (weights and feature characteristics) to the target domain $\mathcal{D}_t$. Transfer learning from natural image datasets, particularly ImageNet [11], using standard large models and corresponding pre-trained weights, has become a de-facto method for cross-domain deep learning applications. Similarly, we use the pre-trained model[12] and transfer the knowledge from their original dataset and weights into our image dataset.

MobileNetV2 is a light weight pre-trained convolution neural network and has been claimed to surpass its predecessors [12]. MobileNetV2 expects images to be of size $\{128*128*3\}$. We use MobileNetV2 as our base model and transfer the learning. The model was trained on millions of images from the Imagenet[11] dataset. MobileNetV2 serves as a feature extractor for our classification model.

The base model consists of more than 100 layers and over 2.5 million parameters and trained over millions of images. We freeze the base model and add a new custom head for our classification. If we do not freeze the base model then the new run would recompute initial weight and parameters hence defeating the purpose of using such a powerful pre-trained model. The last layer of MobileNetV2 outputs $\{4*4*1280\}$ tensor. The output size is not useful for our classification problem. We treat the output from MobileNetV2 as our source domain. MobileNetV2 provides a very efficient mobile-oriented model that can be used as a base for many visual recognition tasks [12].

TABLE I: Cross Domain TL Model - VM-1

Layer (type)	Output Shape	Layer
MobileNetV2 Layers	..	..
out_relu (ReLU)	(4 x 4 * 1280)	Existing
Global Average Pooling	1280	New
Fully Connected Layer	3	New

We flatten the output to fit our classification problem. In order to flatten the output without loosing the weights, we use global average layer. This layer functions similar to max pooling layer where the highest pixel values are transferred to the next layer. Instead, the Global average Layer averages the layer there by maintaining the semantics of learned feature

Authorized licensed use limited to: University of Maryland Baltimore Cty. Downloaded on November 07,2020 at 23:59:00 UTC from IEEE Xplore. Restrictions apply.

weights. Global Average Layer reduces the output from the base model to a vector size of 1280. We then add a drop out, and a fully connected layer to the base model, and create our prediction model for three classes. We use Softmax as an activation function. Table I summarizes addition of custom layers and their types. Algorithm 1 outlines our approach in using Cross Domain Transfer Learning.

Algorithm 1 Cross Domain Transfer Learning

```
Input: Real Time Images from Flood Potential Site
Output: Flood Label: Flood, No Flood, Minor Flood
1: Get Rainfall Intensity at $t_i$
2: Label training image $\mathcal{Y}_i$ into one of the class-Labels based on rainfall intensity
3: Start with the PreTrained Model(s) of choice
4: Freeze the base Model and transfer learned parameters to custom head model
5: Align output of the Base Model to required vector length
6: Add a drop out and a fully connected (dense) layer + Softmax Classifier
7: Validate Model Performance
8: Save Model Weights for Real Time Image Classification
```

V. End to End Memory Network

A simple RNN carries over memory from many previous steps, and hence their implementation becomes cumbersome and resource-intensive [3]. Specially for a chatbot, recalling information from the distant past is not very critical. Chatbot is essentially a question and answering system so it only needs to remember the last statement and the flow. Hence we opted for a variation of encoder decoder model called End to End Memory Network.[15]. Memory network helps chatbot by capturing the most important fact from the provided information.

In memory network, the input sentences are broken down into word vectors and treated as a bag of words (*Input memory representation*). Thus, each sentence(fact) becomes a group of individual vectors $\{x_i\}$. Formally, given an input sentence $\mathcal{X}$ containing words $\{x_1, x_2, \ldots, x_i\}$, memory networks vectorize the words and store them in memory vectors $\{m_i\}$ of dimension d computed by embedding each $\{x_i\}$ of dimension d. This results an embedding matrix $\mathcal{A}$ of size $(d * V)$. The exact same procedure is repeated for the question sentence q set yielding another embedding matrix $\mathcal{B}$ of same size as $\mathcal{A}$. The internal state of the queries is vectorized into u. Now the only remaining part is to find the most relevant answer-sentence to the encoded question. This is measured using the cosine distance or dot product between memory vector $\{m_i\}$ and question's internal state vector u.

$$(p)_i = Softmax(u^T m_i) \ where$$

$$softmax(z)_i = \frac{exp(z_i)}{\sum_j exp(x_j))}$$

and $(p)_i$ is the probability vector. Similarly the output of the sentences are represented into c_i and into a matrix $\mathcal{C}$. The resulting response vectors from the output memory o is summed over the transformed inputs c_i

$$o = \sum_{i=1}^{i} p_i * c_i$$

Finally, to get the prediction for a single question/answer case the sum of the output vector o and the input embedding u is passed through a final weight matrix $\mathcal{W}$ (of size $V * d$) and a softmax to produce the predicted label:

$$o = softmax(\mathcal{W})(o + u)$$

A very related yet more complex method is proposed by [2] called attention model. The working of the encoder decoder model/attention model is depicted in Figure 3. At a high level the attention mechanism helps the next step in recurrent neural networks to find the most relevant words mostly based on the dot product (distance metric) and softmax over it. The concept of context vector is analogous to the memory vector in end to end memory network[15].

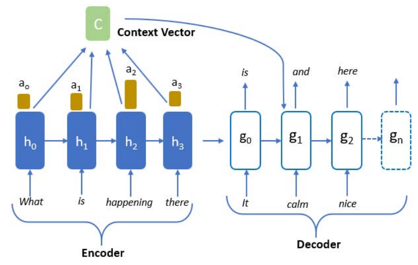


Fig. 3: Attention Mechanism

Attention mechanism ensures that the words we want to focus on are kept as is, and the unnecessary words are discarded.

We highlight these two topics as they are the core concepts in FloodBot's successful implementation. Simply put, we use transfer learning to make sense of scene and use end to end memory model to teach Floodbot to response to user queries.

Algorithm 2 End to End Memory Network

```
Input: User Queries about the event and scene context
Output: Chatbot Response(s): Answer
1: Collect and enrich Question and Answering Corpus
2: Encode Question and Answer into Word Embedding
3: Compute Memory Vector from the word Embedding
4: Multiply memory vector with word embedding
5: Perform dot product memory and softmax them to find the most relevant fact
6: Repeat {2,3,4,5} for each question $q_i$
7: Compute the context vector $c_i$
8: Perform stacked softmax on output and questions internal memory vector
9: Return the most relevant answer
```

VI. Data Collection

We drive our process based on three data sources, the images captured by field camera, the real-time weather data API, and the dialogue corpus. Data is assimilated into their own databases and made available for deep learning models.

Authorized licensed use limited to: University of Maryland Baltimore Cty. Downloaded on November 07,2020 at 23:59:00 UTC from IEEE Xplore. Restrictions apply.

TABLE II: Image to Weather Data

Video_time	Frame_Id	Summary	Icon	PrecipIntensity	PrecipType	Temperature	Humidity
1/25/2020 11:00	frame3971.jpg	Possible Light Rain	rain	0.0735	rain	34.41	0.94
1/25/2020 11:00	frame3972.jpg	Possible Light Rain	rain	0.0735	rain	34.41	0.94
1/27/2020 10:00	frame5739.jpg	Mostly Cloudy	partly-cloudy-day	0.0014	snow	31.07	0.88
1/27/2020 10:00	frame5740.jpg	Mostly Cloudy	partly-cloudy-day	0.0014	snow	31.07	0.88

A. Flood Image Database

The FloodBot implementation starts with real-time images acquisition. These images are transferred into the cloud for pre-processing.The Flood Image database contains images captured in various weather conditions for more than a month. Figure 4 shows the sample images captured during various ambient conditions from the testbed.

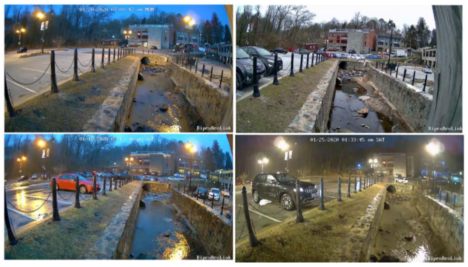


Fig. 4: Flood-Watch Camera

B. Weather Database

During same deployment time frame, we also collected the weather data from an public weather application interface. We find the weather in the area at the time we captured those images. The weather data API allows us to extract weather at that particular location based on latitude and longitude of our camera. We collect and store these records by minutes in our database.

C. Image to Weather Database

We establish a temporal join (image captured time and weather timestamp) and tie the weather condition of the site to our image database. With Image to Weather Database we are able to observe the weather condition at the site both visually (through images) and from data released by meteorological weather stations via their APIs. Table II summarizes the outcome of our three database.

D. Flood Aware Dialogue Corpus

The last piece of data set that we need for our FloodBot is starting knowledge base or the seed corpus. We experimented with multiple non domain specific Question and Answering corpus (Q&A) from various sources [10, 5, 9, 18]. After performing the complexity and relevancy analysis on these datasets, we used Facebook's bAbI-QA for question answering and text understanding [18] as our seed corpus. This dataset is composed of a set of contexts, with multiple question-answer pairs available based on the contexts. We infused more data into same dataset and in same format. In order to create natural hazard data, we collected Storm Events Database from NOAA's National Weather Service (NWS) storm database and transformed into bAbI like Q&A dataset. We show the examples of these dataset in Table III below.

TABLE III: Sample Q&A Corpus

ID	Event_Narrative	Question	Answer
1	Thunderstorms produced heavy rain. Multiple low water crossings were closed.	What Produced heavy rain?	Thunderstorms
2	Local roads, city streets and multiple lanes were impassable. A car was stuck in street flooding"	What is stuck in flooding?	Car
3	A landslide occurred on Hwy 66. There was significant damage to Hwy 66.	What happened to Hwy 66?	LandSlide

VII. Experiment And Evaluation

In this section, we outline the experiment and setting for three Deep Learning Models. We deployed three deep learning models to create FloodBot. Two vision models (V-M1 and V-M2) that enable the field reconnaissance of flooding and possible hazard, and the language model (L-M1) enables human-FloodBot conversation.

A. Vision Model 1

***Rule Based Bulk Image Annotation**:* We call this a rule-based supervised learning because the annotations for training image dataset were provided using the current rainfall intensity. We have collected the weather data synchronous to the images. Thus we programmed database queries and python package to label the image into one of the classes based on recorded rainfall intensity during the image frame timestamp. For example, a nice sunny day with no rain recorded would yield 'No Flood.' Light rain would cause minor floods. Significant rainfall would classify an image as 'Flood'. We categorize the flood images into three classes **No Flood**, **Minor Flood** and **Flood**. The classes are based on turbulence, turbidity and observed high velocity of the flowing water predominantly caused by rainfall intensity. This is a clever labeling technique that is robust and practical, given the large data set.

This is our domain-specific Flood Categorization Model. In order to classify the current flooding situation at the site, we labeled 15,686 images in our training set. We used the automated rule based bulk Image annotation technique discussed above to label 322 images as 'Flood', 2440 as 'minor flood' and 13,125 as 'No Flood' images. This a highly imbalanced dataset so we re-sampled the images to balance the class distribution uniformly. To achieve a more balanced dataset we recreated the images from 'flood' class by copying, altering label data (up-sampling to 500 images) and randomly

Authorized licensed use limited to: University of Maryland Baltimore Cty. Downloaded on November 07,2020 at 23:59:00 UTC from IEEE Xplore. Restrictions apply.

sampled images from other two classes (down sampling to 500 images).

B. *Vision Model 2 V-M2*

This is a vanilla grade SSD implementation of the Single Shot Multi box Object detection used to detect objects ad enrich our chatbot knowledgebase We pass the identified objects as an bag-of-words to the memory network model. The output from this model enriches our FloodBot's knowledgebase to have a meaningful conversation about the flood location. SSD generates scores for the presence of object category by a drawing and bounding box around the detected object.

C. *End to End Memory Network*

The End to End Memory Network Model enables FloodBot to have a conversation with the human . We implemented attention based end to end memory network proposed by [18]. The dataset contains language corpus in three tuples {story: multiple sentences, Question: a single sentence and the Answer}. There are many variations within the dataset but two of the categories are relevant to our model. We implemented two bAbI model types, the first one (yes/no) answering dataset and the second one is two fact question answering system.We train our language model with a predefined generic dialogue data set (non-domain specific) and enrich them with a domain-specific conversation.

VIII. Results

In this section we present results from our deep learning models and their inter dependencies and relevancy in creating AI Conversing Chatbot. All our experiments were ran using Google Colaboratory a GPU powered online machine learning platform.

A. *Vision Model 1*

We trained The transfer learning based V-M1 for 100 epochs with custom head. The model is able to achieve accuracy of 97.18%. It took us about 2.5 hours to train the model on 15,686 images. Figure 5 shows the result from one of our validation set. In the given example, the model correctly classified the image set into {*No Flood, Minor Flood and Flood*} from respectively.

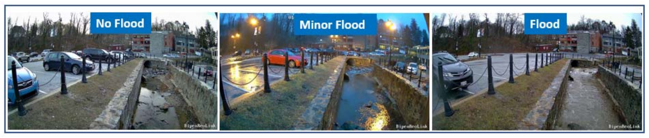


Fig. 5: Results: VM1

1) Baseline Methods: To understand V-M1's comparative performance, we build two kinds of classification models: a shallow network using support vector machine (SVM) and a deep convolutional neural network (CNN). We then train our model on images without using transfer learning. Due to the resource constraints, we sampled 500 images instead of 15,686 from each of the flood categories and extracted low-level image features (1 channel-pixel values) from them and ran through baseline models.

TABLE V: Baseline Model Comparison

Model	**Type**	**Parameters**	**Accuracy**
SVM	RBF Kernels	$\gamma = 0.001, \mathcal{K} = 5$	53%
CNN	4 Layer	relu, softmax	67%
V-M1	Transfer Learning	Glb Avg Pooling,relu,Dropout	97%

Using the non-transfer learning based models we show that the SVM based RBF model is only able to achieve 53% accuracy where as the CNN based model achieves 67% . It is a known fact in computer vision that there needs to be lot of low level image pre-processing before achieving acceptable results from shallow learning model such as SVM. The Cross-Domain Transfer learning immediately improves the accuracy because earlier layers within the base model has already performed and learned feature properties of millions of images.

B. *Vision Model 2*

We randomly selected 2000 image frames and ran them through the V-M2. Our motivation to identify objects in the vicinity is to assess the potential damage should a flood occur. From the dataset collected over a month, the V-M2 model

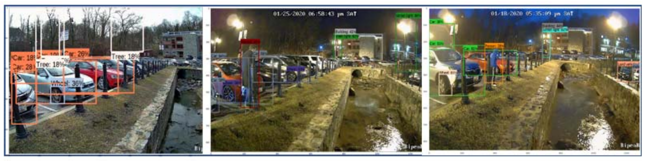


Fig. 6: Results: VM2

identified 90 distinct objects such as {*Car, Wheel, House, Tree, Building, Vehicle, Land vehicle, Tire, Window, Plant, Bench,Truck*}.Percentage wise summary of most and least occurring objects detected from 2000 input image frames by V-M2 is shown in Table VI.

TABLE VI: Most and Least Detected Objects

Objects:	Car	Tree	vehicle	Watercraft	Bicycle
Percentage:	20.79%	17.74%	16%	0.55%	0.40%

Since the camera is pointing to a parking lot area and images are triggered by the motion mostly caused my moving vehicles, the most of the objects detected in 2000 randomly sampled images are those of car and trees.

C. *Model Fusion V-M1 and V-M2*

We show the results of joint learning from V-M1 and V-M2 in Table IV for one of the sample image frame (e.g. *Frame80*). This is one of the sample image frames form our full dataset. The output from V-M1 i.e, current Flooding condition as '*Minor Flood*' is the output from our V-M1 and detected objects {building, Car etc.} are the output of V-M2. Once we have these two elements together (frame by frame and at a given time), the FloodBot has current contextual information to converse and answer queries about real time hazard potential.

D. *End to End Memory Network & Chatbot Implementation*

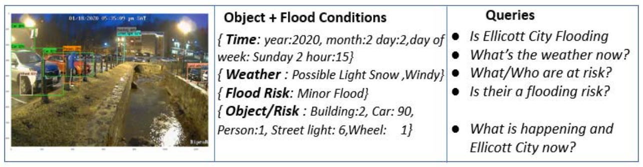
We motivate the results of chatbot implementation by recalling the basic type of chatbot system, a **Closed Domain Retrieval Chatbot**. These chatbots can only converse around

Authorized licensed use limited to: University of Maryland Baltimore Cty. Downloaded on November 07,2020 at 23:59:00 UTC from IEEE Xplore. Restrictions apply.

TABLE IV: Joint Model Learning VM1 & VM2

				VM1	VM2	
frame_id	**summary**	**precipIntensity**	**humidity**	*Flood_Label*	*Object*	*Total_Object*
frame80.jpg	Possible Light Snow and Windy	0.0257	0.81	Minor Flood	Building	2
frame80.jpg	Possible Light Snow and Windy	0.0257	0.81	Minor Flood	Car	90
frame80.jpg	Possible Light Snow and Windy	0.0257	0.81	Minor Flood	Person	1
frame80.jpg	Possible Light Snow and Windy	0.0257	0.81	Minor Flood	Street light	6

information that is stored in the database. The learning and outcomes from our vision models and output is depicted in Figure 7. We have broken down the domain knowledge required by the FloodBot into their own sub domains. The FloodBot can respond to the sub domains{*Time, Weather, Flood Risk, and Objects at Risk*}. For example the answer to "Is Ellicott City Flooding" can be easily answered by 'yes/no question' based on the Flood Risk within that time domain. The overall requirement of the chatbot implementation does not change, so we still need to break the user request into two main parts: utterance and intent. In the examples discussed above, utterance is the sentence/phrase and the user intent is to find the current flooding risk. With this background we now digress into actual deep learning based Chatbot Implementation Results.



Object + Flood Conditions	Queries
{ **Time**: year:2020, month:2 day:2,day of week: Sunday 2 hour:15}	• Is Ellicott City Flooding
{ **Weather** : Possible Light Snow ,Windy}	• What's the weather now?
{ **Flood Risk**: Minor Flood}	• What/Who are at risk?
{ **Object/Risk** : Building:2, Car: 90, Person:1, Street light: 6,Wheel: 1}	• Is their a flooding risk?
	• What is happening and Ellicott City now?

Fig. 7: Floodbot: Q & A System

Our chatbot implementation is based on end to end memory network [18]. We infused additional data into the bAbI corpus and retrained our model. The end to end memory is built using encoder decoder functions from Keras Sequential model API. The model was trained for 120 epochs, with each epoch limited to event narrative max length of 75 words. The maximum query length was set to 6 words. We trained our model with 10,000 event narrative which consists of 9000 original stories from bAbI and 1000 new ones from our flood related narrative. We tested our model on the same composition i.e 900 from original corpus and 100 is flood related. We maintained the original batch-size of 32. We were able to achieve 89% accuracy after infusing our data.

TABLE VII: Memory Network Comparison

Model :	BaseLine(bAbI)	After Data Fusion
Accuracy:	98.55%	89%

The original (baseline) model without the data fusion achieves 98.55% accuracy. While there is a reduction in the original accuracy the achieved accuracy is still within acceptable range. We believe that the reduction in accuracy is due to increase in vocabulary size and disjoint data sets. The accuracy could increase if we use the dataset from same domain with similar context.

E. Model Fusion V-M1 and V-M2 and L-M1

The final output from our deep learning model stack is the integration of all three models. In this section, we present two question and answer result from our FloodBot in Q&A Demo section. The first question is an example from data fusion from NOAA's website and the second example is from the the joint learning V-M1 and V-M2 model infused into Q & A knowledgebase corpus. FloodBot answers based on the decoder portion of End to End Memory network and word embedding.

Chatbot Q&A Demo

Story: {'Six', 'inches' ,'water' , '.' 'Flooding', 'reported','in' , 'town'}
Question:{ 'How', 'deep', 'is', 'water', '?'}
FloodBot's Answer: {Six}

Story: {'Minor', 'Flood' ' . ' , 'Ninety','cars' 'Two' 'Building'}
Question:{ 'Is', 'there flood risk', 'in', 'Ellicott City', '?'}
FloodBot's Answer: {yes-minor}

In the second, test scenario, the answer provided by the FloodBot is a result of model fusion from three deep learning model stack.

F. Model Run Time Metrics

One of the needs for our application is to be quick. After all our models were trained and validated, we saved the model weight as a static file. Running an end to end inference on single image and asking a question to chatbot takes less than a minute. The longest time spent in a single image model inference is by the SSD image regeneration step. The lag seems to be to create (re-writing) the image pixel with bounding box. The bounding box and image reproduction is only needed for Qualitative analysis. During the regular runs we persist the object and score in our database.

IX. Discussion

A. Latency Over Accuracy

The object detection count and score from the vision model V-M2 is probabilistic and somewhat heuristic in nature. We argue that the ultimate use of this kind of application is either to have a sense of emergency or try to estimate the possible damage. Thus the object detection and count measured are mostly for estimation purposes. During disaster and life saving events the timely information provided by our model could be more valuable than trying to get to the correct count.

Authorized licensed use limited to: University of Maryland Baltimore Cty. Downloaded on November 07,2020 at 23:59:00 UTC from IEEE Xplore. Restrictions apply.

B. Federated Deployment & Edge Computing

Currently the images are transported over 3G network for our models inference. This causes some latency in real time FloodBot message propagation and conversation stream. There is no need of big computing power after the initial training is completed and the learned model weights and parameters are served (saved). The learned models can be federated to edge devices. Therefore, a better approach would be to have a Federated Deployment with an end to end system running on an edge computing device such as Raspberry Pi, Jetson Nano or Coral AI.

X. Conclusion

In this paper we presented our approach in creating a FloodBot and infusing multiple data sources. We believe that there could be extensive uses of the proposed application. Chatbot and conversational AI is an active research area and there is a lot of innovation happening in the area. Successful implementation of topics discussed in this paper will make chatbots more robust and suitable for a broader range of applications. In par with the current development in the computer vision research areas, we believe that we have created robust deep learning vision model stack however, our language model and the FloodBot's conversational power is still work in progress.

In our future work, we plan to extend the natural language component of this work and enhance our Floodbot's chatting capacity. This work and architecture of this deployment can be traced using our FloodBot's Twitter account (@umbc_FloodBot). The AI Conversing power and the social media integration of FloodBot into Twitter verse is our ongoing research.

References

[1] Stanislaw Antol et al. "VQA: Visual Question Answering". In: *CoRR* abs/1505.00468 (2015). arXiv: 1505.00468. URL: http://arxiv.org/abs/1505.00468.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2014. arXiv: 1409.0473 [cs.CL].

[3] Mathias Berglund et al. "Bidirectional Recurrent Neural Networks as Generative Models - Reconstructing Gaps in Time Series". In: *CoRR* abs/1504.01575 (2015). arXiv: 1504.01575. URL: http://arxiv.org/abs/1504.01575.

[4] Antoine Bordes, Y-Lan Boureau, and Jason Weston. *Learning End-to-End Goal-Oriented Dialog*. 2016. arXiv: 1605.07683 [cs.CL].

[5] Pawel Budzianowski et al. "MultiWOZ - A Large-Scale Multi-Domain Wizard-of-Oz Dataset for Task-Oriented Dialogue Modelling". In: *CoRR* abs/1810.00278 (2018). arXiv: 1810.00278. URL: http://arxiv.org/abs/1810.00278.

[6] Chaitanya K. Joshi, Fei Mi, and Boi Faltings. "Personalization in Goal-Oriented Dialog". In: *CoRR* abs/1706.07503 (2017). arXiv: 1706.07503. URL: http://arxiv.org/abs/1706.07503.

[7] Ryan Kiros, Ruslan Salakhutdinov, and Richard S. Zemel. "Unifying Visual-Semantic Embeddings with Multimodal Neural Language Models". In: *CoRR* abs/1411.2539 (2014). arXiv: 1411.2539. URL: http://arxiv.org/abs/1411.2539.

[8] Ankit Kumar et al. "Ask Me Anything: Dynamic Memory Networks for Natural Language Processing". In: *CoRR* abs/1506.07285 (2015). arXiv: 1506.07285. URL: http://arxiv.org/abs/1506.07285.

[9] Cristian Danescu-Niculescu- Mizil and Lillian Lee. "Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs." In: 2011.

[10] Pranav Rajpurkar et al. "SQuAD: 100, 000+ Questions for Machine Comprehension of Text". In: *CoRR* abs/1606.05250 (2016). arXiv: 1606.05250. URL: http://arxiv.org/abs/1606.05250.

[11] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[12] Mark Sandler et al. "Inverted Residuals and Linear Bottlenecks: Mobile Networks for Classification, Detection and Segmentation". In: *CoRR* abs/1801.04381 (2018). arXiv: 1801.04381. URL: http://arxiv.org/abs/1801.04381.

[13] Iulian Vlad Serban et al. "Hierarchical Neural Network Generative Models for Movie Dialogues". In: *CoRR* abs/1507.04808 (2015). arXiv: 1507.04808. URL: http://arxiv.org/abs/1507.04808.

[14] Lifeng Shang, Zhengdong Lu, and Hang Li. "Neural Responding Machine for Short-Text Conversation". In: Beijing, China: Association for Computational Linguistics, July 2015, pp. 1577–1586. DOI: 10.3115/v1/P15-1152. URL: https://www.aclweb.org/anthology/P15-1152.

[15] Sainbayar Sukhbaatar et al. "Weakly Supervised Memory Networks". In: *CoRR* abs/1503.08895 (2015). arXiv: 1503.08895. URL: http://arxiv.org/abs/1503.08895.

[16] Oriol Vinyals and Quoc V. Le. "A Neural Conversational Model". In: *CoRR* abs/1506.05869 (2015). arXiv: 1506.05869. URL: http://arxiv.org/abs/1506.05869.

[17] Oriol Vinyals et al. "Show and tell: A neural image caption generator". In: *Computer Vision and Pattern Recognition*. 2015. URL: http://arxiv.org/abs/1411.4555.

[18] Jason Weston et al. *Towards AI-Complete Question Answering: A Set of Prerequisite Toy Tasks*. 2015. arXiv: 1502.05698 [cs.AI].

[19] Tiancheng Zhao et al. "Generative Encoder-Decoder Models for Task-Oriented Spoken Dialog Systems with Chatting Capability". In: *CoRR* abs/1706.08476 (2017). arXiv: 1706.08476. URL: http://arxiv.org/abs/1706.08476.

[20] Fuzhen Zhuang et al. *A Comprehensive Survey on Transfer Learning*. 2019. arXiv: 1911.02685 [cs.LG].

Authorized licensed use limited to: University of Maryland Baltimore Cty. Downloaded on November 07,2020 at 23:59:00 UTC from IEEE Xplore. Restrictions apply.