CNNs for Precipitation Estimation from Geostationary Satellite Imagery

Paul M. Aoki
CS 231N project report
12 June 2017
aoki@acm.org

Abstract

The ability to detect and estimate rainfall from near-real-time satellite imagery is important in many disciplines. In this project,[1] *I extend recent (2016-2017) hydrometeorology work on applying deep learning to this problem by applying modern convolutional neural network (CNN) techniques.* [2] *After outlining the problem, I briefly discuss key prior work in both deep learning and hydrometeorology. I outline the properties of the experimental datasets. Following a discussion of the machine learning methods applied (including those of the prior work), I report my experimental results on the detection and estimation tasks. I conclude with an assessment of these results (which confirm and improve upon the prior work) and give some thoughts on future directions.*

1. Introduction

While there are many different ground and space sensor technologies used to measure rainfall, passive observational data from geostationary satellites is still the only long-term, consistently-calibrated, near-realtime, near-continuous, globally-available data source available. Naturally, there are many research and operational systems that use this data to identify and estimate rainfall on a global basis.

These remain difficult problems, particularly due to temporal and spatial skew. The basic problem of rainfall estimation is often broken down into (1) a *detection*, or binary rain/no-rain classification, problem (where a key challenge is the class imbalance – global hourly frequency of remotely-detectable rain is $< 5\%$) and (2) a (conditional) *estimation* problem (where a key challenge is skew – allowing rare, but important, heavy rainfall events while remaining consistent with prior distributions).

Surprisingly little work has been done to apply modern deep learning techniques in this domain. The notable exception is recent work at U.C. Irvine (published in 2016-2017) that has begun to apply deep learning to infrared imagery from NOAA's GOES satellite constellation [25, 27, 26]. In this research, I (1) replicate the UCI work as a baseline; (2) apply modern convolutional architectures to improve upon them; and (3) provide experimental results supporting the effectiveness of these improvements.

2. Related Work

There are many different modeling problems related to precipitation. For example, it is common to try to build models to predict rainfall at a given point location based on weather station observations, considered as a time series. Examples of this in the neural network literature include [30] and [8].

However, the simple point-measurement approach obviously leaves out a vital spatial aspect – rain clouds, and hence rain, have both extent and paths in space. Data from space-borne radars, microwave sounders, and visible/infrared imagery capture this and are often fused to compensate for their different spatiotemporal coverage and atmospheric penetration properties (see [18] for a useful survey). Well-known operational and research products such as CMORPH [14] and TMPA [12] involve complex, hand-crafted fusion/calibration pipelines that apply relatively simple modeling techniques (histograms, linear regression, etc.) that are easily interpretable. However, there are also examples of shallow neural networks being used as a component of such pipelines [28]. For example, the U.C. Irvine PERSIANN system [11] has been in operational use for nearly two decades, and similar shallow neural networks continue to be developed [17].

More recently, there have been attempts to apply deep learning to meteorology problems. For example, Grover et al. applied deep learning to wind patterns [6]. More closely

[1]**Honor code statement**: This project is the sole work of the listed author, done exclusively for CS 231N. The models described here do not include any third-party software beyond what is distributed with the frameworks listed.

[2]**Scientific validity**: This research is preliminary, and any comparisons to or reproductions of related work have **not** been confirmed with those authors; do not cite or quote without contacting the author.

related is the work of Shi et al. [21] and Klein et al. [16] to process weather radar “imagery” (ground radar displays, essentially screen captures). The main results of which I am aware in applying deep learning to satellite imagery for precipitation detection/estimation are a recent series of papers by Tao et al. [25, 27, 26] that applied a simple 3-layer fully-connected architecture with greedy layer-wise pre-training based on stacked denoising autoencoders [29] to GOES infrared satellite imagery [1].

A key potential benefit of the approach is simplicity. The fact that a simple architecture using single-platform data can produce results comparable to complex, multi-platform pipelines such as that in PERSIANN-CCS [9] is exciting. The goal of the research reported here has been to confirm this research and then extend it using modern architectural ideas.

3. Data

I first replicated the dataset of Tao et al. [25]. This consists of infrared satellite imagery from the NOAA GOES-13 satellite [1], hereafter *GOES*; the National Weather Service (NWS) Quantitative Precipitation Estimate hourly analysis product [19], hereafter *QPE*, as “ground truth”; and the U.C. Irvine PERSIANN-CCS hourly product [9], hereafter *PCCS*, as a state-of-practice baseline.

The dataset covers the *study area*, which is a portion of the Great Plains region (30°-45° N latitude, 90°-105° W longitude). This area is of practical interest as the Great Plains region generally sees large (mesoscale) clusters of intense convective storm systems [10] in the late summer.

3.1. NOAA GOES Imagery

I downloaded 196 GB of raw hourly imagery [1] for infrared bands 3 (WV, $\lambda = 6.5\mu$m) and 4 (IR1, $\lambda = 10.7\mu$m) from the NOAA CLASS repository for summer 2012 and 2013 and winter 2012-2013 and 2013-2014.[3] Raw pixels were extracted from NOAA GVAR AREA files, calibrated,[4] resampled to a ~8 km geographic grid (0.08° $\times$ 0.08°) from ~4 km resolution pixels, navigated[5] and cropped to produce rasters covering only the study area (Figure 1(a),(b)).

3.2. NWS QPE Stage IV

QPE Stage IV [19] is an operational data product that is released in various production stages and at varying time resolutions. The ~4 km resolution pixels in their native HRAP grid (a polar stereographic projection) were resampled to a ~8 km (0.08° $\times$ 0.08°) geographic grid (Figure 1(c)).

3.3. UCI PERSIANN-CCS

PCCS is a research data product based on cloud-cover imagery [9]. I downloaded ~40 GB of compressed data matching the GOES period above. The ~4 km resolution (0.04° $\times$ 0.04°) pixels were resampled to a ~8 km (0.08° $\times$ 0.08°) geographic grid.

3.4. Data Handling and Basic Statistics

Following [25], the first summer/winter were pooled and split into train:validation (75:25), while the second summer/winter were pooled and used exclusively for testing. A summary of the “ground truth” data (Table 1) shows that only a small fraction of the pixels (i.e., pct$_R$) are rainy, leading to severe class imbalance in both seasons. For reasons that are unclear, the summer QPE and GOES data is far more subject to missing hours, and I only include hours for which all of the datasets are available.

Note that the entire study area dataset is $>$ 300 million images; however, the images are 15×15 ($1.2° \times 1.2°$) windows around each pixel in the study area (Figure 2), so each image actually overlaps with a large number of nearby im-

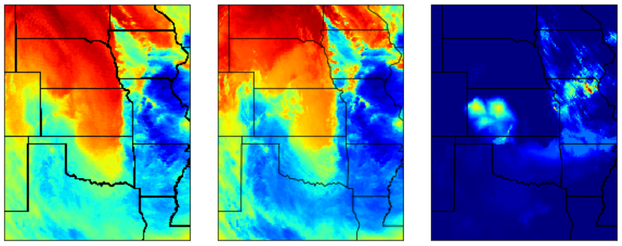

(a) GOES 6.5μm (b) GOES 10.7μm (c) QPE

Figure 1: Example GOES and QPE data for the entire Great Plains study region (30°-45° N, 90°-105° W), using false-color to highlight large summer storm systems. (a),(b) are inputs and (c) is the “ground truth” rainfall rate. Lower values (dark blue) in the input imagery roughly correspond to higher values (rainfall in mm/h) in the labels. Each patch is a 15x15 window within the depicted region.

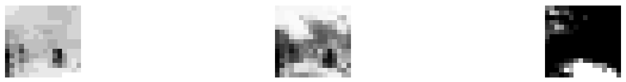

(a) GOES 6.5μm (b) GOES 10.7μm (c) QPE

Figure 2: Example input (GOES) patches (a),(b) and “ground truth” (QPE) patch. It is difficult to see, but there are isolated rainy pixels at the top of (c).

[3]Note that “summer” means June-July-August (JJA) and “winter” means December-January-February (DJF). These are standard meteorology conventions, but this also avoids two anomaly periods (May 2012, Sep. 2013) during which the GOES-13 imager was not operating correctly.

[4]Raw pixels require calibration due to variation across instruments and intra-instrument variation across time.

[5]Although the satellite itself does not change nominal orbital position, the raw GOES pixels are not georeferenced, and the geographic location of a given pixel can vary considerably due to small maneuvers and changes in spacecraft attitude.

ages. After a naive image extraction process filled up a 1 TB file system, I stored the study area data (GOES, QPE, PCCS) as HDF5 files and wrote a Python generator to extract 15x15 images on the fly (essentially, a custom data augmentation pipeline; see Section 5.1.1).

4. Methods

In this section, I describe the main algorithmic techniques (architectures, loss functions, evaluation metrics) applied while obtaining the experimental results of Section 5. (Section 5 will describe the software infrastructure and configuration)

4.1. Architectures

I considered three main architectures (see Figure 3), all of which are intended to learn directly from the (mean-shifted) input pixel values. It should be understood that network trained for detection and estimation may have different "heads" (e.g., with sigmoid or ReLU activation) and that networks with similar architecture are trained separately for the two tasks.

4.1.1 Fully-connected (FC) architecture

In all of their work to date, the UCI authors used a fully-connected architecture. (Structure and parameters are shown in Figure 3(a),(b).) Surprisingly (for very recent work), they used a greedy layer-wise pre-training algorithm to initialize this network – specifically, the *stacked denoising autoencoder* (SDAE) technique [29] with 40% corruption rate. Autoencoders are networks trained to reproduce their outputs using a capacity-constrained representation; SDAEs are autoencoders where the successive layers are trained using some kind of input corruption, most commonly one similar to dropout [24]. The fully-connected (FC) layers use ReLU activations. When using 2-channel inputs, they *concatenate* two stacks that are (to oversimplify a bit) trained and fine-tuned separately (Figure 3(b)) [27].

season	source	total (px)	mean (mm/h)	mean$_R$ (mm/h)	max$_R$ (mm/h)	pct$_R$ (%)
S'12	PCCS	75431250	0.0768	2.5580	74.3224	0.0300
S'12	QPE	75431250	0.0499	1.8515	82.9759	0.0270
W'12	PCCS	96643125	0.0826	1.6094	52.1655	0.0513
W'12	QPE	96643125	0.0370	1.0148	59.9985	0.0364
S'13	PCCS	73305000	0.1325	3.4460	76.7664	0.0385
S'13	QPE	73305000	0.0723	1.7762	120.3330	0.0407
W'13	PCCS	95934375	0.0753	1.4666	69.8254	0.0514
W'13	QPE	95934375	0.0257	0.6339	76.7304	0.0405

Table 1: Statistics for PCCS (example operational product) vs. QPE ("ground truth" product). For the rainy-condition statistics, "rainy" is defined as ≥ 0.1 mm/hr.

An architecture of this kind can exploit spatial information within the area covered by its inputs. However, the ability to generalize some types of information is limited in a non-convolutional approach.

4.1.2 Simple CNN (SCNN) architecture

Use of a convolutional neural network (CNN) is motivated by the desire to consider translationally-invariant features. An "ultra-deep" network is surely not needed for such small images. As such, I selected a "Simplicity"-style architecture [23] that reflects several current design trends: a number of 2D convolutional layers organized into sub-stacks (each with a common filter depth d, and where each sub-stack uses small 3x3 filters, some with stride 1 and some with stride 2 in place of maxpool layers), ReLU activations, and frequent batch normalization [13] layers. (Structure

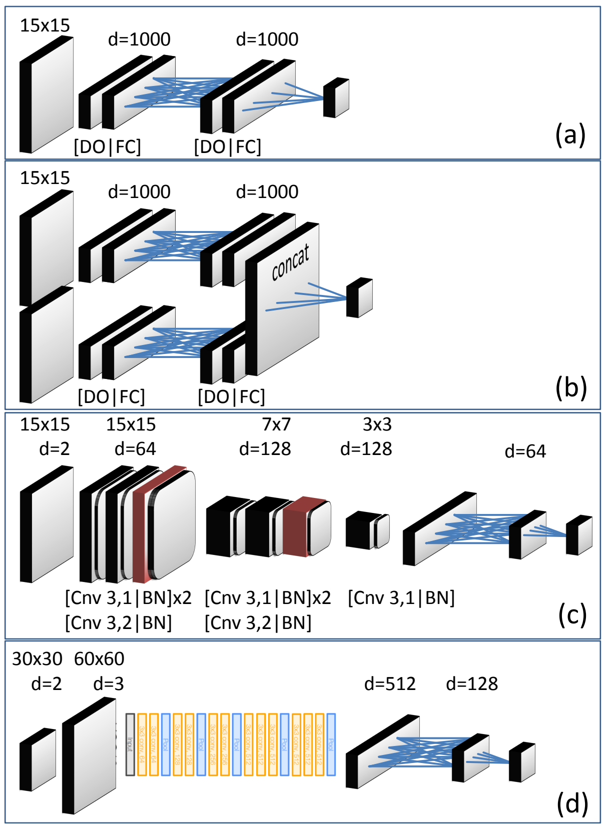


Figure 3: Network architectures discussed in this report. Each includes some fully-connected (FC) and dropout (DO) layers, and may include convolutional (Cnv2D) and batch normlizatios (BN) layers as well. (a) 1-channel FC architecture (as in [25, 26]). (b) 2-channel FC architecture (as in [27]). (c) Simple CNN architecture. (d) Transfer CNN architecture (based on VGG16 [22]).

and parameters are shown in Figure 3(c).) I made this architecture somewhat less simple by replacing the 1x1 convolutional layers with a few dense layers (this seemed to work better in ad hoc experimentation) and adding a spatial dropout [24] layer (with 50% dropout rate) between the 15x15 and 7x7 stacks for regularization.

Note that the network in [23] was designed for CIFAR-10 and uses an initial filter depth of $d = 96$ (with the second sub-stack having $d = 192$). Since this is a different dataset and task, probably simpler than CIFAR-10, the capacity can be expected to be reduced (see Section 5.1.1).

4.1.3 Transfer CNN (TCNN) architecture

The final architecture considered here is an application of transfer learning. A VGG16 [22] network, pre-trained on ImageNet's 224x224 images, is adapted to take an upsampled version of this dataset (the smallest acceptable input image is 60x60). A simple binary classification head built from fully-connected layers is attached to and then trained on top of the frozen VGG16 layers. (Structure and parameters are shown in Figure 3(d).)

While there is no reason to believe that the (computationally expensive) added depth of VGG16 will help with these tasks, it is straightforward to try and it is useful to understand whether this kind of "out of the box" solution turns out to work better than a custom network.

4.2. Loss Functions

Since the tasks here are detection and estimation, the choices of loss function are relatively straightforward. The main complications relate to the class imbalance problem.

4.2.1 Detection

For the (binary) detection task, I use the common *binary cross-entropy* (or sigmoid cross-entropy). Depending on the season, the positive class (i.e., rain present) in the study region may range from 1% to 6% of the total pixels. I experimented with *sample class balancing* (specifically, oversampling the positive class to produce equal numbers of examples[6] and *class weight-balancing*. The former does not require any changes to the loss function, but the latter scales the loss contribution of each instance of a given class by a specified weight. (This functionality is implemented in the framework I use; see Section 5.)

[6]Data augmentation would have been preferable to simple replication, but most methods were rejected for one reason or another. For example, methods such as shifting and cropping were not used as the goal here is to classify the center pixel of the patch. Colormap transformations were not used because this is calibrated infrared emissions data. Rotation remained a possibility.

4.2.2 Estimation

For the (continuous) estimation task, I use the common *mean squared error* (MSE). As with detection, an imbalance (skew) strategy can affect the loss function or not. Here, I again experimented with sample class balancing. However, I tried using an additive, scaled *Kullback-Leibler divergence* term. (This idea is borrowed from [26].) K-L divergence is generally written as:

$$D_{KL} = \sum_c P_{true}(c) \cdot \frac{P_{true}(c)}{P_{pred}(c)}$$

where P_{true} and P_{pred} are the probability mass functions of the classes within the true and predicted values y_{true} and y_{pred}. The loss then becomes $\alpha \cdot D_{KL} + MSE$ for some relative weight α.

4.3. Evaluation Metrics

Like the loss functions, the evaluation metrics vary by task.

4.3.1 Detection

For detection, I use binary *accuracy*, which considers true negatives (TN), and *critical success index* (CSI, a metric common in meteorology) [20], which does not. CSI is similar to the F_1 score from information retrieval:

$$CSI = \frac{TP}{TP + FP + FN}, \quad F_1 = \frac{2TP}{2TP + FP + FN}$$

Both metrics are necessary since either by itself can fail to capture certain types of classification failure. While redundant, we will also occasionally report *false alarm rate* (FAR) and *probability of detection* (POD) for comparison with prior work:

$$FAR = \frac{FP}{TP + FP}, \qquad POD = \frac{TP}{TP + FN}$$

4.3.2 Estimation

MSE also served as the metric as well as the primary loss function.

5. Experiments

I now turn to the experimental results. After describing experimental aspects common throughout this section, I present quantitative detection results; quantitative estimation results; and a qualitative discussion of their implications.

5.1. Common configuration aspects

5.1.1 Software and hardware infrastructure

All experiments were conducted using Keras 2.0.4 [3] on top of TensorFlow 1.1.0 [2], with logs captured and visualized in TensorBoard (see, e.g., Figure 4). All training and experiments were run on a desktop PC with Ubuntu 16.04 and a NVIDIA GP102-based GPU card.

Extension features of many Keras APIs needed to be used. For example, I added (a) custom loss and metric functions (to include K-L divergence estimation and terms[7]); (b) custom batch generator functions (to avoid materializing all patches on disk, and reduce disk I/O overhead); (c) code to programmatically modify the Keras graph.

Greedy layer-wise pre-training is implemented by programmatically modifying and partially re-training the Keras graph, just as one does with transfer learning. The SDAE corruption layers are implemented using a standard Dropout layer that is turned off during later training phases. In Keras, a dropout layer can be turned into a no-op by setting the dropout rate to 0.

[7]Note that while `keras.losses` does provide an implementation of K-L divergence, the function requires its inputs to be valid probability mass functions. Hence, estimating K-L divergence for each minibatch requires implementing a custom loss function that computes density estimates for the minibatch y_{true} and y_{pred}. Here, I compute histograms with 12 bins.

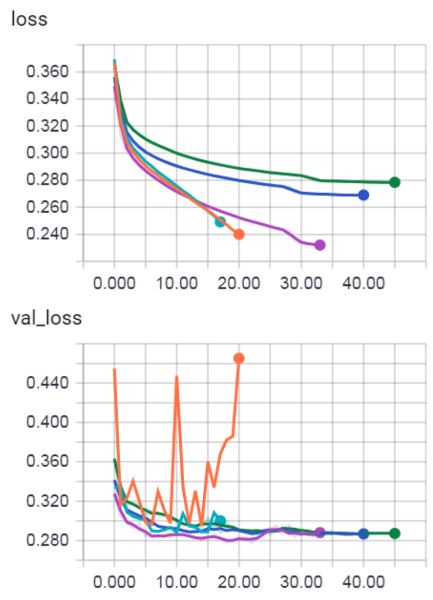


Figure 4: Example SCNN capacity experiment on 10^6 training samples. For the detection task, d =**192** and d =**96** (chosen in [23] for CIFAR-10) clearly overfit, leading to early stop; lower values, such as d =**24** and d =**32**, do not. After additional tests of sensitivity to training set size, d =**64** was used for both detection and estimation tasks.

5.1.2 Parameter search space

The following were used consistently in the results reported in this section (as I could not exhaustively explore all combinations of parameters): (a) FC and CNN kernels used either Xavier normal initialization [5] (before sigmoid activations) or He normal initialization [7] (before ReLU activations); (b) a single adaptive optimization algorithm, Adam [15], was used throughout with default parameters [3], with model training was capped at 500 epochs (applying a learning rate stepdown of 0.1 per 5 epochs of loss non-improvement and early stop at 10 epochs of loss non-improvement); (c) training and testing was done on random subsets of 10^7 patches (out of 117 M in the test set) and a batch size of 10^4 patches.

The very large batch size was used for two reasons. First, from non-systematic experimentation, batch size stochasticity rarely helped with convergence. (This was not true in every experiment run during this project – only for the experiments actually discussed in this report.) Second, use of the K-L divergence requires a density estimate of the minibatch output; given the highly skewed "true" distribution of rainfall rate, we expect to need several thousand outputs to produce any above 10 mm/hour.

5.1.3 Architecture search space

For comparison purposes, the FC architecture capacity, dropout rate, etc. were always fixed to the values given by Tao et al. [25, 27, 26]. As previously mentioned, I implemented their 1- and 2-channel FC architectures (Figure 3(a),(b)); however, since the 2-channel architecture was always superior, only the latter will be discussed here. Similarly, the SCNN architecture was fixed throughout. Since this dataset obviously differs from CIFAR-10, I reduced the capacity parameters of Springenberg et al. [23] after some initial experiments (e.g., Figure 4).

5.2. Detection

The detection task is complicated by class imbalance: the simple majority-class classifier (always return "no rain") achieves over 94% accuracy, albeit with zero CSI (CSI, like F_1 score, ranges on $[0, 1]$). Naturally, the focus of the detection experiments was largely on how to balance accuracy and CSI.

By way of example, Table 2 shows a comparison of FC, SCNN and TCNN as well as the operational PCCS data product. Each deep learning system is tested in three configurations: no class balancing, class weight-balancing (5:1 positive:negative), and sample class balancing (oversampling the positive class to return equal proportions). Three configurations achieve better CSI than PCCS, but only one (SCCN with weight-balancing) attains the state-of-the-art CSI of 0.306 reported in [27].

While it is promising and scientifically reassuring that the level of performance reported in [27] can be achieved and exceeded, it is puzzling that (a) performance of FC without weight-balancing does not directly replicate that of [27] and (b) TCNN appears to converge to the majority-class classifier, even with regularization (dropout) and balancing. Understanding this is an area of future work.

5.3. Estimation

For the estimation task, the basic architectures given above for detection were modified by removing the final sigmoid activation and replacing it with a ReLU activation (rainfall rate is non-negative). Since the new target labels are continuous and the loss functions are different, the FC and SCNN models were retrained from scratch. (TCNN was dropped due to its poor performance on the detection task.)

As with the detection task, much of the focus was on coping with distribution skew. For example, Table 3 shows a comparison between FC and SCNN with two main types of imbalance strategy: adding a K-L divergence term to the standard MSE loss (conceptually similar to class weight-balancing), and sample class balancing. Again, they are also compared to the operational PCCS data product; in addition, they are compared to a constant estimator that always returns "zero rain," which achieves very good MSE. As the table shows, most configurations do produce estimation performance that improves upon that of the comparison references (PCCS, zero). Further, three of these configurations do so while producing rainfall rate distributions that are not wildly divergent from reality (like that of the zero estimator). Figure 5 illustrates the type of distribution associated with such values of D_{KL}. (By contrast, D_{KL} values of 0.005 produce distributions very similar to that of the zero estimator.)

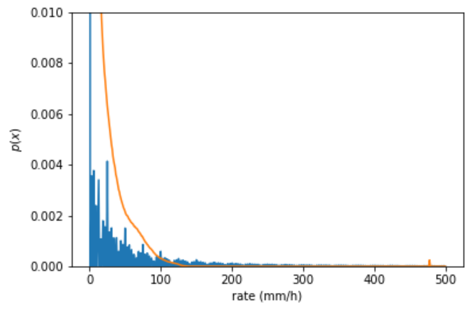
The outlook here is clearer than on the detection task. SCNN was able to produce better estimates (0.381 vs. 0.453) than the current state-of-the-art, even when the latter is tuned for imbalance as well. (The MSE reported in [26] is 1.322, albeit on half of this test set.)

5.4. Qualitative Discussion of Findings

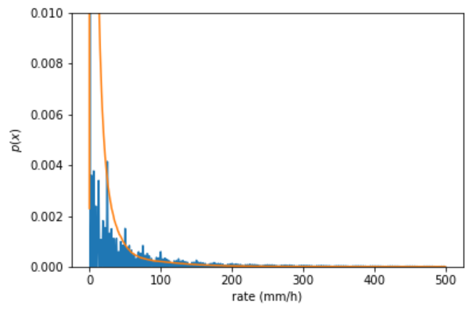
Current operational products such as PCCS are complex fusions of hand-engineered features, with added calibration

	PCCS		
FAR	0.681		
POD	0.391		
CSI	**0.213**		
accuracy	-		
	FC (unbal.)	FC (WB=5:1)	FC (CB=1:1)
FAR	0.404	0.715	0.813
POD	0.109	0.561	0.807
CSI	0.102	**0.233**	0.179
accuracy	0.944	0.893	0.784
	SCNN (unbal.)	SCNN (WB=5:1)	SCNN (CB=1:1)
FAR	0.333	0.490	0.538
POD	0.227	0.443	0.459
CSI	0.204	**0.311**	0.299
accuracy	0.948	0.943	0.937
	TCNN (unbal.)	TCNN (WB=5:1)	TCNN (CB=1:1)
FAR	0.769	0.742	0.742
POD	0.000	0.030	0.030
CSI	0.000	**0.028**	0.027
accuracy	0.949	0.946	0.946

Table 2: Results on the rain/no-rain detection task. Each architecture (FC, SCNN, TCNN) is tested with class weight-balancing (WB) and sample class balancing (CB). While several configurations achieve higher CSI than the operational PCCS system, only SCNN with weight-balancing exceeds the best CSI results [27].



(a) FC, $\alpha = 1$, $D_{KL} = 0.026$.



(b) SCNN, $\alpha = 1$, $D_{KL} = 0.016$.

Figure 5: Illustrative plots of rain rate vs. frequency: **prediction** vs. "**ground truth**." Both models weight D_{KL} and MSE equally ($\alpha = 1$ in Table 3). The "tighter" curve in (b) places more mass on the tails. However, minimizing D_{KL} more strongly is undesirable, as this typically corresponds to all-zero estimates (a vertical orange line at 0).

	PCCS	“zero”	FC	FC	FC	FC	FC	SCNN	SCNN	SCNN	SCNN	SCNN
				K-L	K-L	K-L			K-L	K-L	K-L	
			(unbal.)	($\alpha = 0.2$)	($\alpha = 1.0$)	($\alpha = 5.0$)	(CB=1:1)	(unbal.)	($\alpha = 0.2$)	($\alpha = 1.0$)	($\alpha = 5.0$)	(CB=1:1)
MSE (val.)			0.483	0.466	0.453	0.456	-	0.506	0.506	0.376	0.389	-
MSE (test)	**1.363**	**0.473**	0.476	0.440	**0.453**	0.463	0.573	0.485	0.441	**0.381**	0.401	**0.391**
D_{KL}			0.441	0.073	**0.026**	0.005	0.094	0.441	0.029	**0.016**	0.005	**0.026**

Table 3: Results on the rainfall estimation task. MSE values for sample configurations of SCNN and FC, with corresponding MSEs for PCCS and a constant zero estimator. While most attain a better test set MSE than PCCS, many do not beat zero, and few do so while retaining a realistic distribution of rain rates (see Figure 5 for examples of how D_{KL} values align with the true distribution).

against other sensors due to the fact that imagery cannot directly penetrate clouds. The fact that passive IR-only data can provide comparable task performance (as reported earlier by Tao et al. [25, 27, 26]) is exciting, so confirmation is no small thing. Further, the results here suggest that these results can be improved further.

A first, simple observation is that there is not much evidence that any of the networks – even the deeper networks – are overfitting. All use some form of regularization during training and in general (see, for example, Table 3) the networks seem to generalize acceptably – from training (not shown) to validation and from validation to test.

As an example of how the final results of an architecture like SCNN compares to such operational systems, consider the winter storm shown in Figure 6. This represents a much more challenging scenario than that of Figure 1, as the rain “hot spots” basically correspond to low values in the input imagery – but not over Oklahoma and Arkansas, where there are low values in the inputs but no rain shown in the “ground truth.” PCCS estimates heavy rain there, whereas SCNN estimates at most light rain. However, SCNN is also too tentative in estimating the extent of heavy rain in the northeast of the region (compensating for such low bias tendencies is one reason why PCCS does what it does).[8] Hence, it seems likely to me that additional data layers will be required to match the QPE “ground truth” more closely.

5.5. Potential Directions

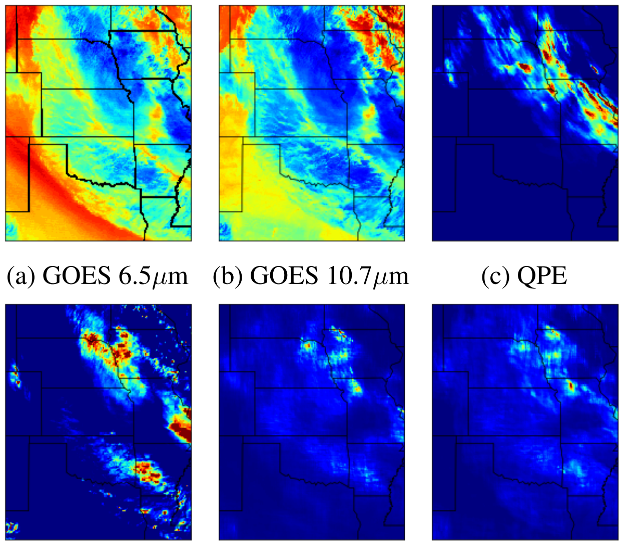
There are several obvious technical directions for improvement:

First, one way to improve detection accuracy without reducing CSI too much might be to apply ideas from semantic segmentation. The detection models tested here tend to converge to the majority-class classifier without balancing; in that configuration, semantic segmentation would probably not be improved by such techniques (which increase spatial coherence – but coherence is not the problem when guessing “always false”). However, balancing tends to add some positive noise; in that configuration, spatial coherence might help. (As a caveat, it is worth noting that, unlike the usual applications of semantic segmentation, it is not unusual to find completely isolated rain pixels in this dataset.)

Second, it might be possible to produce a super-resolution version of the 4 km data using higher-resolution images from low-Earth orbit satellites[9] as a source of

[8]As mentioned in Figure 6, PCCS applies a separately-training R/NR detection model, so its estimate here is actually conditional on the presence of rainfall; the estimate of SCNN is unconditional, and MSE would improve further if the spurious near-zero estimates were filtered out.

[9]LEO satellite data cannot be substituted for GEO satellite data because



(a) GOES 6.5μm (b) GOES 10.7μm (c) QPE

(d) PCCS (e) SCNN, K-L (f) SCNN, bal.

Figure 6: Sample outputs for a storm in the test set (Winter 2013-2014). As with Figure 1, **low** areas in the inputs (a),(b) generally correspond to **high** rain rate areas in the “ground truth” (c) ($>$ 10 mm/hr). The operational system, PCCS (d), skews high and spuriously predicts heavy rain over Arkansas. SCNN with either K-L divergence (e) or class balancing (f) is still biased toward the mean. (Note that PCCS estimates in (c) have had a binary rain/no-rain classifier applied, removing the spurious low but non-zero estimates visible in (e),(f).)

patches (see, e.g., [4]). The point is not to use the super-resolution version for visualization or study per se, but rather to see whether such images provide an "interpretation" of lower-resolution images (i.e., the likely appearance of cloud patterns) that can be useful when considering each given single pixel area where estimation/detection is occurring.

Third, we can take advantage of the fact that we have a time series of images and apply RNNs/LSTMs, as done for ground radar images by Shi et al. [21]. However, the datasets are missing a fair number of hourly observations, making the "time series" irregular as well as widely-spaced in time.

6. Conclusions

In this project, I replicated the experimental setup, models and the performance levels reported in the current state-of-the-art in hydrology. I further tested convolutional architectures under the same circumstances and showed improvement on the basic metrics of the detection and estimation tasks. As mentioned earlier, the fact that good results (comparable to, if not better than, mature operational models such as PCCS) can be obtained – at least, for this scenario – with so few data layers and so little pipeline complexity is quite remarkable.

While the improvements over the state-of-the-art *architecture* (FC) in this application were not enormous, they do at least confirm the prior work (in the case of detection [25, 27]) and do show larger improvements over what is actually *reported* in the literature for somewhat different experimental conditions (in the case of estimation [27]). (The imbalance/skew in the dataset also makes the baseline accuracy/MSE quite high at the start.)

there is not enough of it. A LEO satellite is only able to revisit a given geographic location every few days. However, this may still constitute enough patches to be interesting for the purpose here.

References

[1] GOES N Series Data Book (Rev. D). Technical Report CDRL PM-1-1-03, NASA GSFC, Greenbelt, MD, 2009.

[2] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: A system for large-scale machine learning. In *Proc. OSDI*, pages 265–283. USENIX, 2016.

[3] F. Chollet. Keras. `https://github.com/fchollet/keras`, 2015.

[4] C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In *Proc. ECCV*, pages 184–199. Springer, 2014.

[5] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proc. AISTATS*, 2010.

[6] A. Grover, A. Kapoor, and E. Horvitz. A deep hybrid model for weather forecasting. In *Proc. KDD*, pages 379–386. ACM, 2015.

[7] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proc. ICCV*, pages 1026–1034. IEEE, 2015.

[8] E. Hernández, V. Sanchez-Anguix, V. Julian, J. Palanca, and N. Duque. Rainfall prediction: A deep learning approach. In *Proc. HAIS*, pages 151–162. Springer, 2016.

[9] Y. Hong, K.-L. Hsu, S. Sorooshian, and X. Gao. Precipitation estimation from remotely sensed imagery using an artificial neural network cloud classification system. *J. Appl. Meteorol.*, 43(12):1834–1853, 2004.

[10] R. A. Houze. Orographic effects on precipitating clouds. *Rev. Geophys.*, 50(1), 2012.

[11] K.-L. Hsu, X. Gao, S. Sorooshian, and H. V. Gupta. Precipitation estimation from remotely sensed information using artificial neural networks. *J. Appl. Meteorol.*, 36(9):1176–1190, 1997.

[12] G. Huffman, R. Adler, D. Bolvin, G. Gu, E. Nelkin, K. Bowman, Y. Hong, E. Stocker, and D. Wolff. The TRMM Multi-satellite Precipitation Analysis: Quasi-global, multi-year, combined sensor precipitation estimates at fine scale. *J. Hydrometeorol.*, 8:38–55, 2007.

[13] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. ICML*, pages 448–456, 2015.

[14] R. Joyce, J. E. Janowiak, P. A. Arkin, and P. Xie. CMORPH: a method that produces global precipitation estimates from passive microwave and infrared data at high spatial and temporal resolution. *J. Hydrometeorol.*, 5:487–503, 2004.

[15] D. P. Kingma and J. L. Ba. Adam: A method for stochastic optimization. In *Proc. ICLR*, 2015.

[16] B. Klein, L. Wolf, and Y. Afek. A dynamic convolutional layer for short-range weather prediction. In *Proc. CVPR*, pages 4840–4848. IEEE, 2015.

[17] V. M. Krasnopolsky and Y. Lin. A neural network nonlinear multimodel ensemble to improve precipitation forecasts over continental US. *Adv. Meteorol.*, 2012, 2012.

[18] K. Kunzi, P. Bauer, R. Eresmaa, P. Eriksson, S. B. Healy, A. Mugnai, N. Livesey, C. Prigent, E. A. Smith, and G. Stephens. Microwave absorption, emission and scattering: Trace gases and meteorological parameters. In *The Remote Sensing of Tropospheric Composition from Space*, pages 153–230. Springer, 2011.

[19] Y. Lin and K. E. Mitchell. The NCEP stage II/IV hourly precipitation analyses: Development and applications. In *Proc. AMS Annual Conf. Hydrology*, 2005.

[20] J. T. Schaefer. The critical success index as an indicator of warning skill. *Weather & Forecasting*, 5:570–575, 1990.

[21] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-C. Woo. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Proc. NIPS*, pages 802–810, 2015.

[22] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proc. ICLR*, 2015.

[23] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. Technical Report arXiv:1412.6806, 2014.

[24] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 15:1929–1958, 2014.

[25] Y. Tao, X. Gao, K. Hsu, S. Sorooshian, and A. Ihler. A deep neural network modeling framework to reduce bias in satellite precipitation products. *J. Hydrometeorol.*, 17(3):931–945, 2016.

[26] Y. Tao, X. Gao, A. Ihler, K. Hsu, and S. Sorooshian. Deep neural networks for precipitation estimation from remotely sensed information. In *Proc. CEC*, pages 1349–1355. IEEE, 2016.

[27] Y. Tao, X. Gao, A. Ihler, S. Sorooshian, and K. Hsu. Precipitation identification with bispectral satellite information using deep learning approaches. *J. Hydrometeorol.*, page to appear, 2017.

[28] F. J. Tapiador, C. Kidd, K.-L. Hsu, and F. Marzano. Neural networks in satellite rainfall estimation. *Meteorol. Appl.*, 11:83–91, 2004.

[29] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *JMLR*, 11:3371–3408, 2010.

[30] P. M. Williams. Modelling seasonality and trends in daily rainfall data. In *Proc. NIPS*, pages 985–991, 1997.