# Lightweight Deconvolutional Neural Networks for Efficient Cloud Identification in Satellite Images

John Merriman Sholar
Department of Computer Science, Stanford University
jmsholar@cs.stanford.edu

## Abstract

*Many applications which analyze satellite imagery depend on preprocessing - one important step is the identification of atmospheric contamination (*e.g. *cloud cover) so it can be ignored by downstream applications. Most solutions so far (*e.g. *Hollstein* et al. *[16]) rely on pixel-level classification; we frame the problem as an image segmentation task, and apply deconvolutional neural networks (Noh* et al. *[24]) to identify several classes of atmospheric contamination in Sentinel-2 satellite images. We experiment with models that rely on multispectral data, as well as RGB-only data, which is more generalizable.*

*We use a two-stage process in which we first produce improved training data by using the Sentinel-2 proprietary cloud mask to train pixel-level decision tree classifiers. We then use the output of the decision trees as labeled training data for deconvolutional models. With this setup, we learn a generalized deconvolutional model which is capable of accurate ($\approx .90$ per-pixel accuracy), fast (1M pixels per second) cloud identification using only RGB data, which can be used to conduct this task on any satellite platform.*

## 1. Introduction

### 1.1. Motivation

Recent advances in satellite imagery and computer vision (particularly since the popularization of convolutional neural networks (CNNs) by Krizhevsky *et al.* in 2012 [21]) present opportunity in the intersection of these two fields. We cite as an example the work of Jean *et al.*, who use satellite images to predict poverty levels in remote areas [19]. For these applications, removing noise (especially clouds and other atmospheric contamination) from satellite images is a critical step in using these images to maximum effect. This can be done via mosaicking over time, but for temporally sensitive applications where mosaicking is not possible, an acceptable substitute is to identify and mask out clouds in images, to allow downstream models to ignore them. We develop a series of models for this task.

### 1.2. Task Definition

We frame our task as a semantic segmentation problem. Given a satellite image, we predict a class of atmospheric condition for each pixel. We explore several tasks, each of which is defined by a different set of output classes:

1. 2-class task: {CLEAR, CLOUD}
2. 3-class task: {CLEAR, CLOUD, CIRRUS}
3. 4-class task: {CLEAR, CLOUD, CIRRUS, SHADOW}
4. 6-class task: {CLEAR, CLOUD, CIRRUS, SHADOW, WATER, SNOW}

We present in figures 1 and 2 example input and output for this task. In figure 1, the model in question (a baseline cloud mask developed by the European Space Agency for its Sentinel-2 satellites) is successful in identifying the clouds in the image. In figure 2, it fails to recognize cirrus clouds.

We track several metrics for evaluation, including cross-entropy loss for probabilistic models, per-pixel accuracy for all models, and inference time (measured in pixels per second) for all models. Cross-entropy loss is computed by applying the softmax function to predicted logits for each output class

$$\text{Softmax}(x)_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

and taking the cross-entropy loss of the Softmax output $q$ with respect to the ground-truth distribution $p$ (a one-hot vector indicating output class)

$$J_{\text{Cross-Entropy}}(p, q) = -\sum_x p(x) \log q(x)$$

Accuracy is measured naively by number of pixels correctly classified over total number of pixels, or more robustly via multiclass F1 score. Inference time is included to account for the fact that satellite data is generated very quickly, and that fast models are necessary to keep up with preprocessing demand.

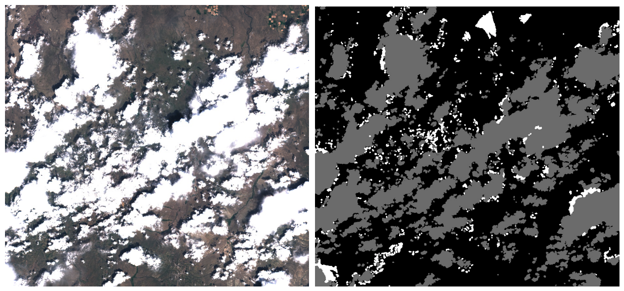

Figure 1: Sample cloud identification input (left) and output (right). For output, black indicates clear sky, white indicates cirrus or semitransparent clouds, and gray indicates opaque clouds. (Source: Sentinel-2)

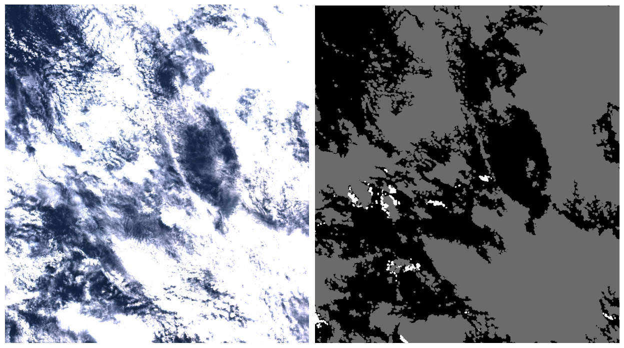

Figure 2: An example of poor cloud identification. As in the previous example, black indicates clear sky, white indicates cirrus or semitransparent clouds, and gray indicates opaque clouds. Note that while almost all of the image is obscured by semitransparent clouds, large portions are marked clear. (Source: Sentinel-2)

Finally, we note that difficult evaluation will be a theme throughout this paper. The Sentinel-2 cloud mask is inherently noisy data, which means that even if a model is performing exceptionally well, it will still be penalized by the poor labels provided by the Sentinel-2 cloud mask. This is also true in the case where we use the output of our decision tree pixel-level classifier as labeled input for our convolutional models. Thus, though we provide quantitative evaluation metrics for the accuracy of our models, analysis must also be strongly qualitative and visual, especially because the ultimate goal of the model is to mask out unuseful information and preserve useful information - an inherently subjective task.

## 1.3. Historical Methods and Related Work

We survey related work in both cloud detection in satellite images and image segmentation in a broader context.

### 1.3.1 Remote Sensing Cloud Detection

To date, methods to identify clouds in satellite imagery have been fragmented, platform-specific, and heavily dependent on human-engineered features. We cite as an example the work of Hallahan and Prepperneau [12], who employ a feature they describe as the "RGB values corresponding to the darkest pixel of the coastal aerosol band" for Landsat-8 cloud detection - other historical methods such as those of Irish *et al.* [18] and Gao *et al.* [7] demonstrate similarly overspecific features. Some more robust models, such as those of Hagolle *et al.* [10] [11] or Zhu *et al.* [30] have relied on temporal methods which analyze the same location over time and use changes in appearance to detect cloud cover. For temporally sensitive applications, such methods are also infeasible.

Those models that achieve state-of-the-art accuracy (such as the proprietary cloud mask for the Sentinel-2 dataset [5], seen in figures 1 and 2, or the expansion of the Fmask algorithm by Zhu *et al.* [29]) rely on years of human feature engineering. Even then, these models do not perform well in all cases (as seen in figure 2), do not perform well with few spectral bands (*e.g.* RGB only), and do not generalize to new satellite platforms. Perhaps one of the most important challenges in this field at this point is the development of a robust generalized cloud mask algorithm - one which relies only on RGB bands and is thus extensible to any new satellite platform.

### 1.3.2 Deep Semantic Segmentation

Almost all models developed to-date rely on pixel-level linear classification or other methods with limited predictive power, such as decision trees. Neural networks, by contrast, have the potential to learn highly nonlinear decision boundaries and eliminate tedious human feature engineering; CNNs in particular have the ability to take advantage of spatial covariance in satellite image data. Having surveyed current efforts in remote sensing, we turn to a survey of image segmentation, in an attempt to marry the two.

Current state-of-the art methods for semantic segmentation include basic deep convolutional networks (employed by Chen *et al.* [2] in conjunction with conditional random fields), fully convolutional networks (Long *et al.* [23]), and deconvolutional networks (Noh *et al.* [24]). Also of interest are models originally intended for object detection (which are frequently applicable to segmentation to some extent) such as the R-CNN model developed by Girshick *et al.* [8]. We adapt both fully convolutional and deconvolutional models for our implementations.

Other techniques that have been shown to be effective in image segmentation include atrous (dilated) convolution [3], which we also employ to expand receptive fields for large cloud structures. Also of interest are the methods of

Chen *et al*. [4], who discuss methods for scale-aware semantic segmentation, which also prove valuable for large cloud structures.

Finally, given the importance of inference speed in this task, we examine the methods of Iandola *et al*. [17], who develop SqueezeNet - a model with AlexNet-level accuracy on the ILSVRC dataset [26] with orders of magnitude fewer parameters and faster inference. We also examine the work of Han *et al*. [14] [13], who pursue similar research aimed at improving model efficiency while maintaining performance.

## 1.4. Data

### 1.4.1 Sentinel-2 Proprietary Cloud Mask

We curate a dataset of satellite images from the Sentinel-2 satellite platform [5]. The Sentinel-2 dataset was selected for several reasons. First, it features 13 spectral bands, which provide considerably more information than standard RGB data. Some bands, in particular the $1.375\mu m$ band, colloquially the "cirrus band", are targeted specifically at detection of cloud cover. Secondly, it features 10 meter spatial resolution and 5 day temporal resolution, allowing for crisp images, and temporal analysis over short time spans. Data from Sentinel-2 is initially analyzed and downloaded through Google Earth Engine [9] before it is transferred to a local platform.

We curate a dataset of 60 Sentinel-2 images, each of which is roughly 10,000 by 10,000 pixels. These images are sliced into smaller images of size 224 by 224 pixels ("tiles"), which are split into training, validation, and test data. As a precaution against mingling training and test data, tiles from a single 10,000 by 10,000 Sentinel-2 image are included in exactly one of the sets.

The Sentinel-2 dataset is bundled with a proprietary cloud mask (seen in figures 1 and 2). That is, the European Space Agency (which manages Sentinel-2) calculates its own cloud mask for these satellite images, and distributes this information with the images. Each pixel is labeled using one of three classes: {CLEAR, OPAQUE, CIRRUS}. Although the mask is far from perfect, it performs adequately in expectation, and provides the closest thing to a large labeled dataset for this task. We use this dataset as noisy training data.

### 1.4.2 Hollstein Pixel-Level Dataset

Additionally, when training classifiers that operate on a per-pixel level rather than on larger images, we use a public dataset curated by Hollstein *et al*. [16]. The dataset features 3 million hand-labeled points from Sentinel-2 images. These points are roughly evenly sampled from around the globe and throughout the year, to ensure full climatic coverage. Each point is labeled using one of six classes:

| Dataset | Train Size | Val. Size | Test Size |
|---|---|---|---|
| Sentinel-2 | 60 K Tiles | 30 K Tiles | 30 K Tiles |
| Hollstein | 2.1 M Points | 500K Points | 500K Points |

Table 1: Sentinel-2 and Hollstein dataset statistics.

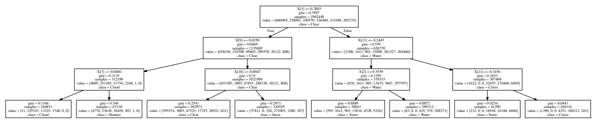

Figure 3: A decision tree of depth 3 learned on the data provided by Hollstein *et al*., for classification of individual Sentinel-2 pixels.

{CLEAR, CLOUD, SHADOW, SNOW, CIRRUS, WATER}. Thus, for per-pixel classifiers, we can support a larger number of more precise classes. However, because these datapoints correspond to labeled points on images, rather than completely segmented images, this dataset cannot be used for training convolutional models.

## 2. Baseline Methods and Results

All baseline model are implemented using the the Python Scikit-Learn library [25]. All code is currently hosted privately on GitHub, but is available upon request.

Our baseline methods revolve around classification for individual pixels. For this task, we use the dataset provided by Hollstein *et al*. described in section 1.4.2.

## 2.1. Decision Tree

We reimplement the methods detailed by Hollstein *et al*. [16] - namely using decision trees for pixel-level classification. The inputs to the decision tree are the 13 spectral band values corresponding to a single Sentinel-2 pixel, and the output is one of the 6 given classes for the Hollstein dataset. We train a decision tree of depth 5 with a minimum impurity split of 0.01 (that is, no node derived in training contains less than 1 percent of the samples in its parent node - a tactic which prevents overfitting). Due to space constraints, we present a decision tree of depth 3 learned on the same data in figure 3.

## 2.2. Fully Connected Neural Network

Our second baseline method attempts to perform pixel-level classification using a Multilayered Perceptron (MLP), also called a fully connected neural network.

Our MLP features two hidden layers, each connected via affine transformation and a RELU nonlinearity. The output of the network is calculated via a final affine transformation

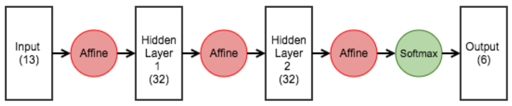

Figure 4: The multilayered perceptron architecture used for individual pixel classification.

followed by a softmax function. This architecture is visualized in figure 4.

After careful attempts to improve upon the performance of the decision trees using our pixel-level MLP (with tuning of layer sizes, regularization constants, and other hyperparameters), we fail to achieve a boost in performance, while significantly increasing inference time compared to that of the decision tree model. This could be a result of improper training or data preprocessing, but is more likely a consequence of properties inherent in the task of pixel-level classification. The information presented by the band values for a single pixel provides limited information and no spatial context - it's highly likely that any separability in the data can be learned using decision trees and does not require the more robust nonlinearities induced by an MLP.

## 2.3. Baseline Results

### 2.3.1 Decision Tree Results

We train decision tree models for both 3-class classification and 4-class classification. We do not explicitly train a decision tree model for 2-class classification, but note that such output can be achieved by combining the CLOUD and CIRRUS labels of the 3-class decision tree into a single CLOUD super-class. We attempt to train models for 6-class classification, but note that the results of these models are inconsistent (and thus unsuitable as input for a convolutional model), and can be improved upon by simply treating the SNOW and WATER classes as CLEAR (yielding a 4-class problem).

We provide sample outputs for the 3-class and 4-class decision trees in figure 5.

### 2.3.2 Baseline Output as Deep Network Input

We have discussed the tenuous reliability of the Sentinel-2 labels, and note that while they serve as noisy training data (frequently wrong but accurate in expectation), they do not provide good evaluation data. Their considerable inaccuracy means that even models which perform excellently are penalized in evaluation.

In an effort to alleviate this, we use our decision tree models to generate a more robust dataset. That is, we train DT models on the Hollstein dataset, and apply these to Sentinel-2 tiles. The output of this primitive segmentation

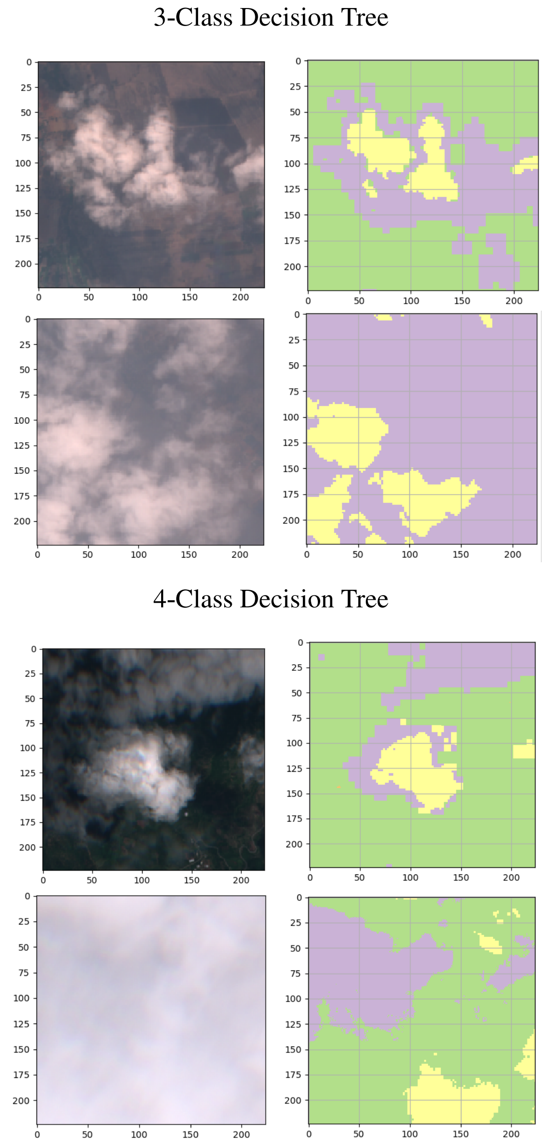

Figure 5: Example outputs of the decision tree classifier. Key: {Green: CLEAR, Yellow: CLOUD, Purple: CIRRUS, Orange: SHADOW}. The 3-class decision tree performs well in expectation, and its outputs (rows 1 and 2) are used as input for convolutional models. The 4-class decision tree struggles to detect the SHADOW class in row 3. In row 4, it is apparent that the introduction of a new SHADOW class has diminished the classifier's ability to label the original three classes.

is used as labeled training data for our convolutional models. Readers may find this methodology initially flawed: it's possible that this will limit our convolutional models to only be as good as our linear models. However, because our convolutional models have the ability to take advantage of spatial covariance in our data, they are in fact capable of significantly outperforming out linear models. In the same way that the Sentinel-2 cloud mask provides noisy training data (frequently wrong but correct in expectation) for training our DT models, the output of these DT models provides considerably more accurate (but still somewhat noisy) training data for our convolutional models. This two-step setup is also advantageous because DT models are trained on the Hollstein dataset, which means we can augment training labels to support all 6 output classes of the Hollstein dataset, rather than the 3 used by the Sentinel-2 proprietary cloud mask.

## 3. Methodology

### 3.1. Pre-Training

It is increasingly common for new convolutional architectures to incorporate a "pre-trained" network as part of their model. We cite as an example Long *et al.* [23], who in their work with deconvolutional networks train a large architecture which first repeatedly condenses an image using convolutional and max pooling-layers, and then re-expands it using a series of transpose convolutional and max-unpooling layers. The entire first half of the architecture (the "convolutional" half) is initialized using weights originally trained by Simonyan and Zisserman [27] for their VGG-16 and VGG-19 models on the ImageNet Large Scale Visual Recognition Challenge [26] (ILSVRC) dataset.

Unfortunately, we find such techniques to be relatively unhelpful for our task. Models pretrained on ImageNet classification data (or even other image segmentation datasets, such as Microsoft Coco [22] or PASCAL VOC [6]) are adapted to the images commonly found in these datasets, which are vastly different from satellite imagery. Additionally, the most notable pretrained models often support only 3 RGB bands – another roadblock in transferring to segmentation of Sentinel-2 images, which have 13 spectral bands. Thus, we do not employ pre-trained models in the development of our architectures.

### 3.2. Transpose Convolution

Both of our models employ transpose convolutional layers, which may be unfamiliar to some readers. We provide a brief overview of these methods.

A standard convolutional layer learns a collection of $n$ filters $f \in \mathbb{R}^{k\times k\times d}$, where $k$ is the length-width dimension of the filter and $d$ (the "depth") is the number of filters in the input. Each filter is convolved over the image. That is, it is unrolled, and its inner product $f^T r$ produces a scalar output $a \in \mathbb{R}$ for some input region $r \in \mathbb{R}^{k\times k\times d}$. By contrast, a transpose convolutional layer develops $n$ filters $f_T \in \mathbb{R}^{k\times k\times 1}$, which are multiplied by a scalar region of the input $r \in \mathbb{R}$ to produce an output of size $a \in \mathbb{R}^{k\times k\times 1}$.

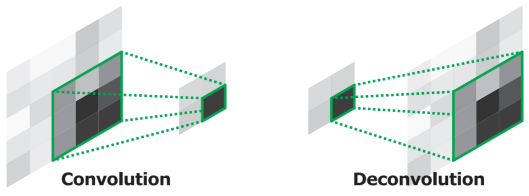

Figure 6: Convolution and transpose convolution. Source: Noh *et al.* 2015 [24].

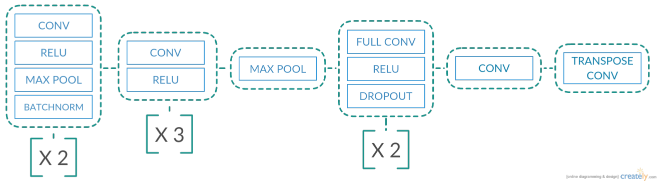

Figure 7: The AlexNet-FCN architecture used for image segmentation.

These outputs are combined via convolution to produce an output image. In this way, transpose convolution can be thought of as an operation to "reverse" convolution, turning one input into many outputs, where convolution turns many inputs into one output. Convolution and transpose convolution are illustrated in figure 6.

### 3.3. Fully Convolutional Networks

For our initial implementation of a convolutional neural network for this task, we implement the AlexNet-FCN model described by Long *et al.* [23]. This model is essentially a decapitated adaptation of the AlexNet architecture implemented by Krizhevsky *et al.* [21], where the final fully connected layers of the original model have been replaced with fully convolutional layers which preserve spatial dimensionality. These are followed by transpose convolution to scale the output of the network up to the size of the original image. This architecture is visualized in figure 7.

With only a single deconvolutional layer, the Alexnet-FCN model learns to identify cloud structures with some success, but produces coarse outputs. We seek to rectify this with the deconvolutional models described in section 3.4.

### 3.4. Deconvolutional Neural Networks

#### 3.4.1 Theory

Deconvolutional networks (Noh *et al.* 2015 [24]) have the potential to improve on the results of fully convolutional networks by employing multiple transpose convolution layers to improve the granularity of output. For the reader's

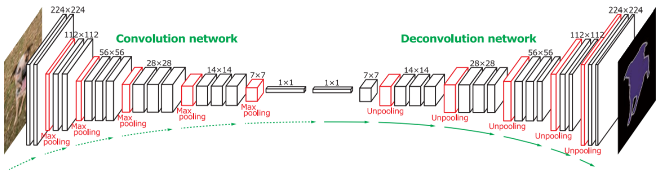

Figure 8: The deconvolutional net originally implemented by Noh *et al.*, on which our model is based. Source: Noh *et al.* 2015 [24].

| Layer Type | Filters | Kernel Size |
|---|---|---|
| Input | # Bands | – |
| Convolutional | 64 | 5 |
| Convolutional | 128 | 5 |
| Fully Convolutional | 256 | 1 |
| Fully Convolutional | 256 | 1 |
| Transpose Convolutional | 128 | 5 |
| Transpose Convolutional | 64 | 5 |
| Scores (Transpose Conv.) | # Classes | 1 |

Table 2: Our Deconvolutional architecture, designed to balance fast inference and robust prediction. All *Conv layers use RELU activations, and have inputs normalized with batch normalization.

reference, we provide a diagram of the original architecture implemented by Noh *et al.* in figure 8. Though our model is considerably more lightweight, the convolutional-deconvolutional structure remains the same.

### 3.4.2 Implementation

We provide our complete implementation of our deconvolutional model in table 2. The output of any convolutional (Conv) or fully convolutional (FConv) layer (but NOT transpose convolutional (TConv) layers) is passed through a RELU activation (popularized by Krizhevsky *et al.* 2012 [21]). The input of any Conv/FConv/TConv layer is normalized using batch normalization (except the input to the first Conv layer, which has already been normalized in preprocessing). The inclusion of these batch normalization layers is found to significantly improve the performance of the network (consistent with the original findings of Noh *et al.* [24]). Noh *et al.* also employ layers of max-pooling and max-unpooling (in which the argmaxes of the corresponding max-pooling step are used to unpool later in the network), but we find this technique has no effect on performance and significantly increases inference time for this task.

We also achieve a significant boost in qualitative performance by weighting our cross-entropy loss function to place greater importance on correct classification of non-CLEAR output classes (noting that a CLOUD false positive is preferable to a CLOUD false negative, as the latter passes bad information to a downstream application). That is, our loss function becomes

$$J_{\text{Cross-Entropy}}(p, q) = -\sum_{x} \alpha_x p(x) \log q(x)$$

Where $\alpha_{\text{CLEAR}} = 0.25$ and all other values of $\alpha_x = 1$.

Our deconvolutional model is trained using the output of the decision tree classifier. The model is implemented using Google's Tensorflow framework [1]. We employ ADAM optimization (Kingma and Ba 2014 [20]) with a constant maximum learning rate of $0.0005$ and $\beta_1 = 0.9, \beta_2 = 0.999$ We find that decaying the learning rate using vanilla minibatch SGD does not improve performance. We train using minibatches of size 20 (where minibatch size is chosen to maximize usage of GPU memory), for two epochs over our training set, at which point we achieve convergence. Training takes roughly 1 hour on an NVIDIA Tesla K40 GPU.

## 4. Results

### 4.1. Evaluation Metrics

We employ several metrics in evaluating our model, including cross-entropy loss between predicted and ground-truth distributions, per-pixel classification accuracy, inference time, and others. Unfortunately, our use of noisy training data (which is frequently wrong but generally correct in expectation) means that though we're able to learn accurate models using inaccurate labels, we are not necessarily able to *fairly* evaluate our models using these inaccurate labels. Further complicating matters is the fact that cloud detection is an inherently subjective task. Our goal is to mask out unuseful parts of the image (clouds), so that downstream applications receive only useful information. However, what a downstream application considers "useful" can vary widely. Though we provide quantitative evaluation metrics in the form of decision matrices, we remind the reader that these metrics are only reliable to a certain degree. A hypothetical "best" model would perform considerably worse than a model that was able to fit perfectly to the noisy labels.

### 4.2. Qualitative Results

For the reasons discussed in section 4.1, evaluation of qualitative results becomes particularly important for this task. We present in figure 9 a compilation of qualitative results. Hard convolutional output is obtained by taking the probability of a given output class and thresholding it above some value $\alpha$ (for figure 9, we use $\alpha = 0.5$). This allows a downstream application to vary $\alpha$ in order to vary what degree of confidence in a predicted CLEAR label is acceptable.

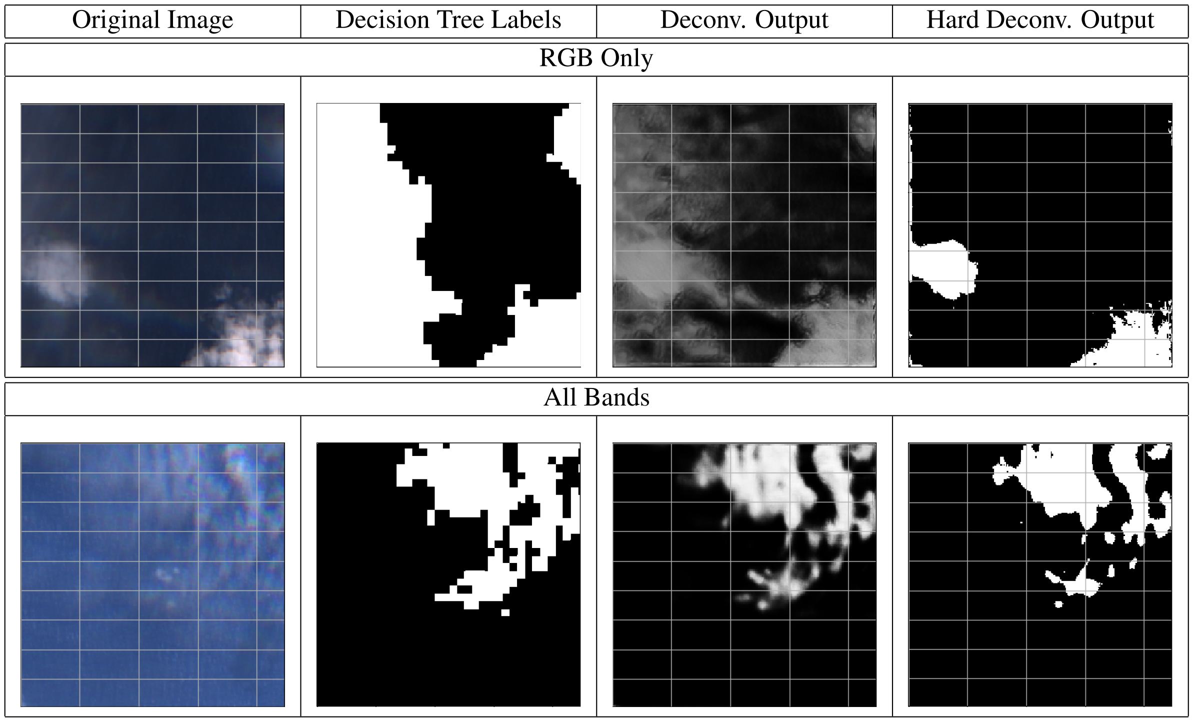

Figure 9: Qualitative outputs of our deconvolutional output model for the 2-class problem with output classes {CLEAR, CLOUD}. The inputs used for this analysis were not used in training or validation. The convolutional outputs for row 1 are calculated using only RGB input data. The convolutional outputs for row 2 are calculated using all 13 Sentinel-2 spectral bands. Decision tree values are always calculated using all 13 Sentinel-2 spectral bands.

### 4.2.1 RGB-Band Input vs. 13-Band Input

Comparison of the outputs of the convolutional models reveals that there is little to no qualitative difference between those outputs generated using all 13 spectral bands and those generated using only the RGB bands. This is somewhat intuitive, as ultimately the model should only mask out atmospheric contamination that impedes visible light (all 10 non-RGB frequencies lie on the edge of or outside the visible spectrum). We note additionally that outputs of the RGB-only model tend to be more robust and conservative (assigning more moderate probabilities for all output classes) than those of the model trained on all 13 bands. This likely signals a need for greater regularization in the 13-band model, which has more information on which to make strong (possibly over-zealous) predictions.

Perhaps most importantly, qualitatively the outputs from our convolutional models improve considerably on those of our decision tree models (which were reimplemented from the recent work of Hollstein *et al*. [16]). We note that qualitatively our results constitute an initial implementation of a robust, generalized cloud-mask which uses only RGB data. Thus, our model can be applied to data from a variety of satellite platforms in addition to Sentinel-2.

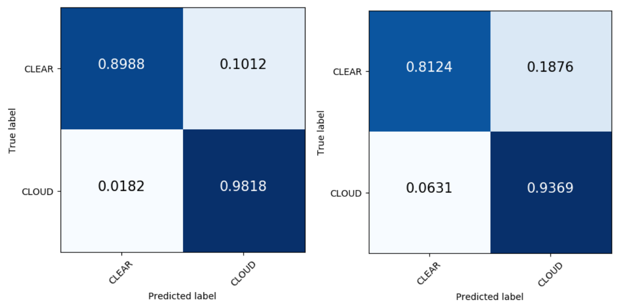

Figure 10: Confusion matrices for our deconvolutional model using all bands as input (left) and only RGB bands as input (right).

## 4.3. Quantitative Results

We present in figure 10 confusion matrices for the output of our convolutional models for the 2-class problem, using the output of our decision tree models as noisy ground truth. As was our stated goal, we achieve an extremely low false-negative rate for CLOUD pixels, at the cost of a somewhat high false-positive rate for CLOUD pixels. Un-

| Model | Inference Speed (Pix/Sec) |
|---|---|
| Hollstein Decision Trees | 16.25 Million |
| Decision Trees | 15.55 Million |
| Deconv. Net. (RGB Bands) | 1.28 Million |
| Deconv. Net. (All Bands) | 1.02 Million |

Table 3: Inference speeds for our models. We include as a reference point for other work in the field the results of Hollstein *et al.*, who also use decision trees for pixel-level classification of satellite imagery.

derstandably, these accuracies decrease further (but remain relatively high) for models that use RGB-only data rather than 13-band data.

Finally, we present in table 3 the inference speeds of our various models. Though our best model suffers a roughly 16x decrease in inference speed compared to state-of-the-art models from the remote sensing academic community, we note its potential for considerably more accurate output than traditional methods, in addition to its potential for increased inference speed with future model optimization.

## 5. Future Work

Our model makes progress towards a generalized cloud mask, but many improvements are needed.

**A More General Dataset** The model was trained, tested, and evaluated on about 120,000 tiles, representing 2000 tiles each from only 60 large Sentinel-2 images. This means the model was trained on dataset with large, highly correlated subsets (that is, all tiles generated from a single image). A more general model can be achieved by training on a smaller number of tiles from a larger number of initial images.

**More Efficient Inference** In an effort to further improve inference time, future work should explore techniques for improving architectural efficiency (reducing number of parameters and inference time while maintaining performance), such as those employed by Iandola *et al.* [17] in their SqueezeNet model, or those employed by Han *et al.* [13] [14].

**Improved Shadow Identification** Currently, our model is able to readily identify CLOUD and CIRRUS classes, but struggles with SHADOW classes. This occurs primarily as a result of a lack of training data (and the failure of our intermediate decision tree models to produce reasonable training data for the 4-class problem). Shadows can also introduce significant noise in downstream applications, and future work should focus on improving in this regard.

## 6. Conclusion

In this work we have shown that lightweight convolutional-deconvolutional neural networks show promise for the task of efficiently identifying cloud cover in satellite images. We accomplish this via a two-step process in which we first generate robust training data using decision trees, and then use this training data to train our deep models. Deconvolutional models are well-suited to this task for their ability to produce smoothly varying, highly granular output, and further research in the intersection of these fields will likely yield improved performance on this task.

## Acknowledgements

We wish to acknowledge the course staff for Stanford's course CS 231N: Convolutional Neural Networks for Visual Recognition, especially Shyamal Buch, for their feedback throughout this research, as well as their provision of funding for Google Cloud Platform compute resources.

We also wish to acknowledge Andre Hollstein and Hannes Diedrich of the German Research Center for Geosciences, who diligently corresponded with us and agreed to share the dataset used in their research.

Finally, we wish to acknowledge the Stanford Sustainability and Artificial Intelligence Lab, and in particular George Azzari and Stefano Ermon, who provided access to their `atlas` cluster and GPU resources, as well as feedback throughout the research process.

## Code Usage

Over the course of our research, we acquired and repurposed significant amounts of code from the following sources:

- Tensorflow's tutorial on training deep convolutional neural networks (`https://www.tensorflow.org/tutorials/deep_cnn`)
- Model implementations hosted by Long *et al.* for models described in "Fully Convolutional Networks for Semantic Segmentation" (`https://github.com/shelhamer/fcn.berkeleyvision.org`)
- Model implementations hosted by Noh *et al.* for models described in "Learning Deconvolution Network for Semantic Segmentation" (`https://github.com/HyeonwooNoh/DeconvNet`)
- Stanford Sustainability and Artificial Intelligence lab members Anthony Perez and George Azzari, who wrote and shared code useful for pulling and preprocessing images from Google Earth Engine.

## References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*, 2014.

[3] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915*, 2016.

[4] L.-C. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille. Attention to scale: Scale-aware semantic image segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3640–3649, 2016.

[5] M. Drusch, U. Del Bello, S. Carlier, O. Colin, V. Fernandez, F. Gascon, B. Hoersch, C. Isola, P. Laberinti, P. Martimort, et al. Sentinel-2: Esa's optical high-resolution mission for gmes operational services. *Remote Sensing of Environment*, 120:25–36, 2012.

[6] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[7] B.-C. Gao, A. F. Goetz, and W. J. Wiscombe. Cirrus cloud detection from airborne imaging spectrometer data using the 1.38 $\mu$m water vapor band. *Geophysical Research Letters*, 20(4):301–304, 1993.

[8] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[9] Google Earth Engine Team. Google earth engine: A planetary-scale geo-spatial analysis platform. `https://earthengine.google.com`, December 2015.

[10] O. Hagolle, M. Huc, D. V. Pascual, and G. Dedieu. A multi-temporal method for cloud detection, applied to formosat-2, ven$\mu$s, landsat and sentinel-2 images. *Remote Sensing of Environment*, 114(8):1747–1755, 2010.

[11] O. Hagolle, M. Huc, D. Villa Pascual, and G. Dedieu. A multi-temporal and multi-spectral method to estimate aerosol optical thickness over land, for the atmospheric correction of formosat-2, landsat, ven$\mu$s and sentinel-2 images. *Remote Sensing*, 7(3):2668–2691, 2015.

[12] N. Hallahan and C. Prepperneau. Cloud detection and removal techniques for landsat 8 imagery.

[13] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[14] S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems*, pages 1135–1143, 2015.

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[16] A. Hollstein, K. Segl, L. Guanter, M. Brell, and M. Enesco. Ready-to-use methods for the detection of clouds, cirrus, snow, shadow, water and clear sky pixels in sentinel-2 msi images. *Remote Sensing*, 8(8):666, 2016.

[17] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[18] R. R. Irish. Landsat 7 automatic cloud cover assessment. In *AeroSense 2000*, pages 348–355. International Society for Optics and Photonics, 2000.

[19] N. Jean, M. Burke, M. Xie, W. M. Davis, D. B. Lobell, and S. Ermon. Combining satellite imagery and machine learning to predict poverty. *Science*, 353(6301):790–794, 2016.

[20] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[22] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer, 2014.

[23] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[24] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1520–1528, 2015.

[25] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[26] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[27] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[28] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[29] Z. Zhu, S. Wang, and C. E. Woodcock. Improvement and expansion of the fmask algorithm: cloud, cloud shadow, and snow detection for landsats 4–7, 8, and sentinel 2 images. *Remote Sensing of Environment*, 159:269–277, 2015.

[30] Z. Zhu and C. E. Woodcock. Automated cloud, cloud shadow, and snow detection in multitemporal landsat data: An algorithm designed specifically for monitoring land cover change. *Remote Sensing of Environment*, 152:217–234, 2014.