

Robustness
of
 Limited Training Data
for
 Building Footprint Identification

cosmiQ
 works®

October 2019

Contents

| | |
|--|-----------|
| Introduction | 2 |
| 1 Performance versus Training Data | 4 |
| 1.1 Motivation: The Utility of Limited Data | 4 |
| 1.2 Background: Learning Curves | 5 |
| 1.3 The Plan: SpaceNet4 Resources | 5 |
| 1.4 The Result: Limited Data Rises to the Occasion | 7 |
| 2 Statistical Methods | 9 |
| 2.1 Training Time | 9 |
| 2.2 Error Estimation | 11 |
| 2.3 Curve Fitting and Extrapolation | 12 |
| 2.4 Putting the Method to the Test | 13 |
| 2.5 A Look at Ensembling | 14 |
| 2.6 Conclusions | 15 |
| 3 Multiple Geographic Locations | 16 |
| 3.1 Method | 16 |
| 3.2 Results and Conclusions | 18 |
| 4 Exploring Geographic Differences | 23 |
| 4.1 Transferability | 23 |
| 4.2 Geography and the Low-Data Falloff | 26 |
| 4.3 Next Step | 29 |
| 5 Architectures Compared, & Final Thoughts | 31 |
| 5.1 Two Model Architectures | 31 |
| 5.2 Project Conclusions | 34 |
| Links | 36 |

Introduction

How much training data is enough?

In many geospatial deep learning problems, training data is a precious resource. With this motivation, we undertake a study of how the amount of training data affects model performance. A better understanding of this relationship can help guide decision-making surrounding data collection and analysis approaches. To focus on a specific problem, this study will be restricted to the challenge of finding building footprint polygons in satellite imagery. While there is no shortage of available building imagery (with the financial costs of imagery and labeling being the primary constraints), buildings may be a proxy for other, scarcer targets. For this study, we draw heavily on the resources of SpaceNet, using SpaceNet’s corpus of high-quality open-source labeled satellite data as well as model architectures submitted to a SpaceNet Challenge.

In the first chapter, we look at a self-contained case study of building footprint model performance versus amount of training data, using data and a model from SpaceNet 4. It is seen that model performance initially rises rapidly as training data is increased, with diminishing returns as the amount of training data is increased further. Chapter 2 delves into statistical methods for analyzing performance versus training data curves, such as error bars and empirical fits. This includes a demonstration of using a small amount of training data to roughly predict performance with much more training data. Chapters 3 and 4 draw on SpaceNet 2 data to expand the analysis to multiple cities. In Chapter 3, we compare performance across four world cities. An argument for generalized models in place of city-specific models is made for a common use case. Chapter 4 takes an exploratory approach to understanding city-specific differences in model performance. After looking at model transferability, we investigate visual differences among cities that may affect model performance in the low-training-data limit. Chapter 5 starts by checking that key results hold up with a different model architecture, noting that models cannot always be ranked by performance without specifying the amount of training data. The final section, 5.2, lists the key takeaway conclusions from this project.

This monograph is based on a series of blog posts published on the CosmiQ Works DownLinQ blog (links found in the endnotes^{1,2,3,4,5}) and a post on the Towards Data Science blog.⁶

Chapter 1

Performance versus Training Data

It's a question that gets asked over and over again: How much data do I need to train my neural network? In this chapter, we will explore that question and answer it in the context of a specific case from the field of remote sensing imagery analysis. We'll show that small amounts of data can perform surprisingly well. Subsequent chapters will look at whether the answer is affected by geography and model architecture.

1.1 Motivation: The Utility of Limited Data

Many different variables determine the ultimate mission impact of satellite imagery. To make sense of it all, CosmiQ Works approaches this topic through the conceptual framework of the Satellite Utility Manifold.⁷ The utility of any given satellite imagery data set depends simultaneously on a number of attributes, each of which warrants careful investigation. Previous DownLinQ blog posts have explored the utility of satellite imagery as a function of its resolution, revisit rate,⁸ number of bands,⁹ and viewing angle.¹⁰ Here, we investigate the utility of an imagery data set as a function of *how much* data it contains.

To define the scope of this analysis, we'll look at the task of building footprint detection in high-resolution satellite imagery, with a ground sample distance (GSD) of less than a meter. We ask the question: is model performance largely data-limited, even with a city's worth of data? Or are there opportunities for meaningful results with even a tiny amount of data? The time and expense of data labeling make this an important issue, so let's find out.

1.2 Background: Learning Curves

Although the importance of data set size for deep learning is widely acknowledged, surprisingly little attention has been given to quantitative analyses of the relationship between data set size and model performance for a fixed model architecture.

Where such research has been done, it has primarily focused on multiclass classification tasks. In a classification context, the term “learning curve” describes a plot of either accuracy or error as a function of training data set size. An early paper on this topic (Seung et al., 1992)¹¹ modeled neural networks as thermodynamic systems — and got published in a theoretical physics journal. One of that paper’s conclusions is that learning curves should scale as a constant plus or minus an inverse power law term. Cortes et al. (1993)¹² highlighted the practical applications of this result, and it was recently put to use by Cho et al. (2016)¹³ in a study of classifying medical images.

However, we have not found any analogously-thorough treatment of F1 scores for instance segmentation, of which building footprint detection is an example. Furthermore, we have not found much treatment of data set size issues in a satellite imagery context. (An exception to the latter, albeit not a deep learning example, is an early CosmiQ Works effort to use machine learning to measure ship headings.¹⁴)

1.3 The Plan: SpaceNet4 Resources

To study the effect of imagery data set size on utility for building footprint identification, we train the same SpaceNet4 competitor model on varying amounts of data. The data sets released through SpaceNet¹⁵ are among the highest-quality open-source labelled satellite imagery, and the associated SpaceNet Challenge competitions result in the development and open-sourcing of high-performing geospatial deep learning models.

For this analysis of data set size, we use the SpaceNet4 data set,¹⁶ with imagery of Atlanta taken from a variety of viewing angles. The imagery is grouped into three categories: “nadir” with viewing angles within 25 degrees of nadir, “off-nadir” with viewing angles from 26 to 40 degrees off nadir, and “far-off-nadir,” with viewing angles exceeding 40 degrees off nadir. Where “overall” performance is reported, this is defined as a simple average of performance for the three categories. Some example imagery is shown in Figure 1.1. A follow-up experiment will look at nadir imagery from cities in the SpaceNet2 data set, to see whether the results are geography-dependent.

As for the model, we use the building footprint detection model that placed fifth in the SpaceNet4 Challenge.¹⁷ Although not the top-performing submission, this model¹⁸ outpaces the winner on inference times by more than a factor of ten. It also has the advantage of a widely-familiar architecture, consisting of

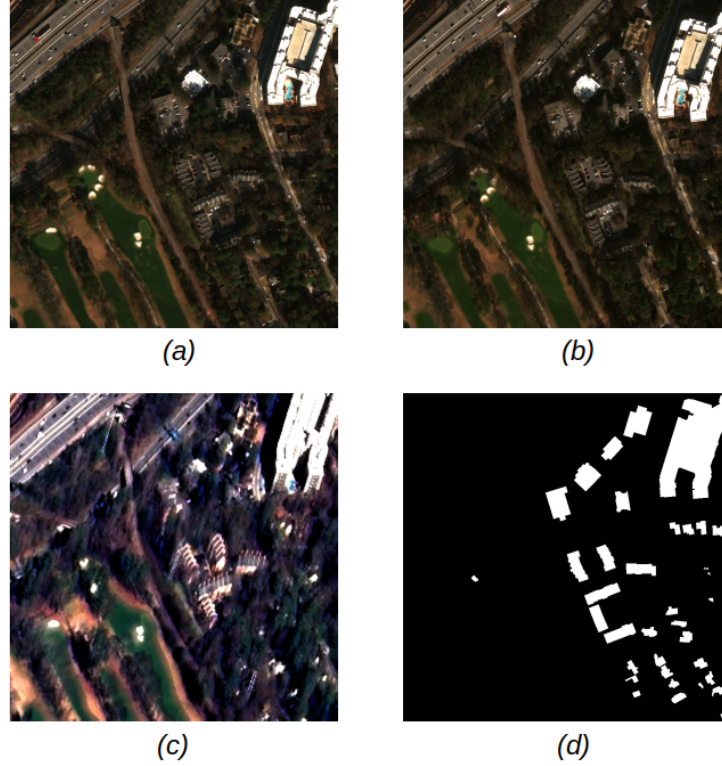


Figure 1.1: (a-c) Different imagery tiles for the same physical location: (a) An on-nadir view, (b) an off-nadir view from the north, (c) a far-off-nadir view from the south; (d) the images’ common ground truth building footprint labels.

a U-Net with a VGG-16 encoder and corresponding decoder. Nevertheless the model, contributed under the screen name “XD_XD,” performs within 5% of the top-performing model overall. Another follow-up experiment will see how the results change if a different model architecture is substituted for this one.

As in the SpaceNet4 Challenge, model performance is evaluated¹⁹ based on F1 score (the harmonic mean of accuracy and precision) for finding building footprints with IoU (intersection-over-union, a measure of overlap with ground truth) above 0.5. The score is evaluated with testing data that is different from the training/validation data.

To expedite the analysis, some modifications are made to XD_XD’s design. Most importantly, the original ensemble of three models is replaced by just one, since this threefold reduction in training time results in a mere 3% decrease in performance.

The model is first trained with the full SpaceNet 4 training data set. This data set contains imagery tiles of 900 by 900 pixels, each corresponding to an area of 450m on a side, for 50cm GSD. The average tile has 63 building footprints. The SpaceNet 4 training data includes 1064 tile locations within Atlanta, with 27 views of each location. Since a quarter of the tile locations are set aside for validation, the number of images actually used for training is $1064 \times 27 \times 3/4$, or about 21,500 images. After training with the full training data set, the process is repeated nine more times, reducing the amount of data by about one-half each time.

1.4 The Result: Limited Data Rises to the Occasion

The results of training the same model with different amounts of training data are shown in Figure 1.2. The key result is immediately evident: Model performance rises rapidly with training data when there is not much data available, but further increases in data provide diminishing returns. The shape of the curves is roughly logarithmic, although a more exact functional form will be developed in the next chapter.

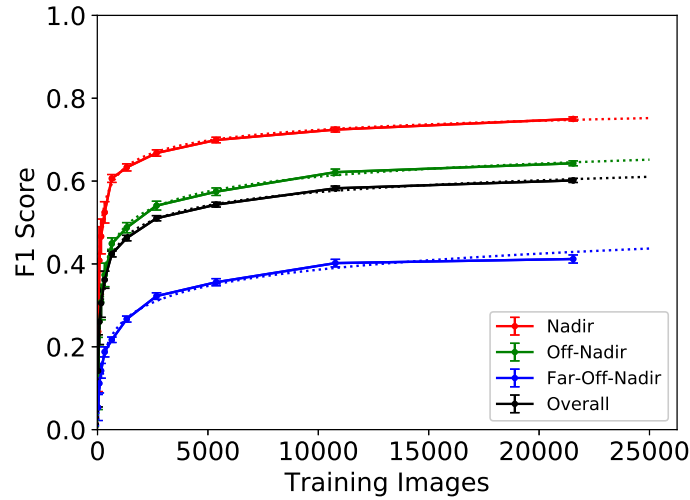


Figure 1.2: Model performance, as measured by F1 score, versus number of images used for training, excluding validation. These images include 27 views of each unique location. Dotted lines are fitted curves. The x-axis is amount of training data, *not* training time.

This means that the model when trained with even a small amount of data performs remarkably well. Compared to using the full data set, using just 3%

of the data still provides $2/3$ of the performance. To take an extreme example, training on the 27 views of the single location randomly selected for use here provides about $2/9$ of the performance of training with the full data set. That's almost a quarter of the maximum performance — with one eight-hundredth of the data.

There's much more to unpack here, including discussions of just where those error bars and fitted curves in Figure 1.2 come from. (Here's a preview: the error isn't constant, and the fitted curves aren't logarithmic!) Look for that, along with matters of training time, ensemble-building, and what it all means, in the next chapter.

Chapter 2

Statistical Methods

In the previous chapter, we asked how geospatial deep learning performance varies with the amount of training data, and we trained a building footprint detector with different amounts of data to see that variation in action. If you haven't read that chapter, go back and take a look now, because next we're going to dive into the details of how it was done and what it all means.

The model's performance as a function of the amount of training data is shown in Figure 2.1 (identical to Figure 1.2 in the previous chapter). Performance is measured by F1 score²⁰ for building footprint IoUs²¹ exceeding a threshold of 0.5, and the number of training images is the number of 450m by 450m tiles used from a 0.5m GSD dataset²² with 27 tiles (at different viewing angles) for each physical location.

As seen in Figure 2.1, the rapid rise of performance with training data when data is scarce contrasts with the diminishing improvements as training data increases further. This means that even small data sets can provide fairly good results. But to better understand what's happening, we need to look more closely, starting with the question of how long it takes to train the model with different amounts of data.

2.1 Training Time

Diving further into the results, we first consider how long it takes for the model to train on different amounts of data. Figure 2.2 shows overall performance versus training time as each model is being trained. Using the full data set, it takes roughly 60 epochs for the model to converge on its optimum performance. Not surprisingly, the process is faster when there's less data to train on. When the training images only number in the hundreds, the model reaches its maximum performance in less time than it takes to run through five epochs with the full data set.

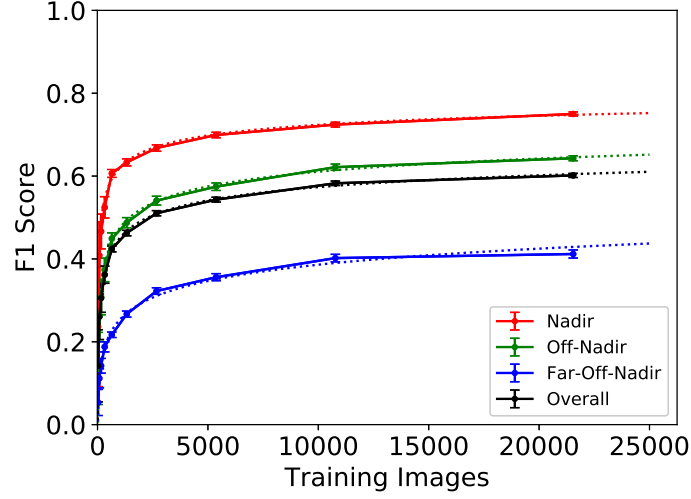


Figure 2.1: Model performance, as measured by F1 score, versus number of images used for training, excluding validation. These images include 27 views of each unique location. Dotted lines are fitted curves. The x-axis is amount of training data, *not* training time.

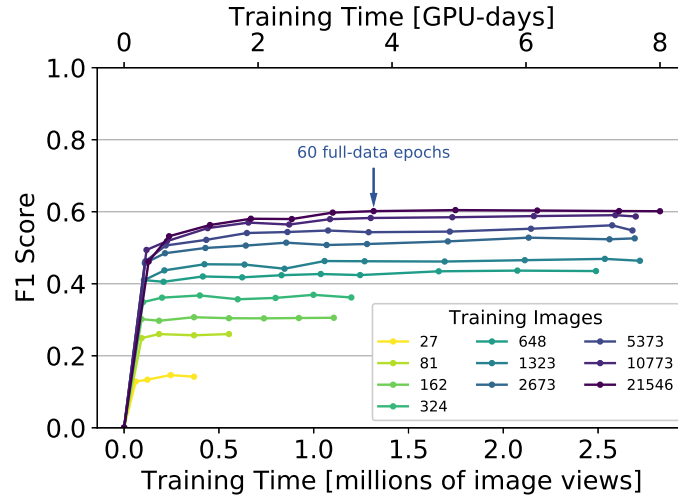


Figure 2.2: Overall F1 score versus training time. Training time is reported in terms of image impressions as well as approximate training time on an Nvidia Titan Xp GPU.

As shown in Figure 2.2, cases with the six largest amounts of training data were each trained for eight GPU-days on Titan Xp GPUs, while the cases with the four smallest amounts of training data were trained less to save computation time. Figure 2.1 reflects the state of each of these cases after a training time of either the maximum training time shown in Figure 2.2 or the equivalent of about 60 epochs with the full data set, whichever is lesser.

2.2 Error Estimation

Simply quoting a machine learning model’s performance metric, without also estimating the uncertainty of the result, is of limited value. Without error bars, it is unknown whether a model with a somewhat higher performance score is significantly better than one with a lower score. Indeed, without error bars it isn’t even clear whether retraining the same model will lead to similar results.

Figure 2.3 shows the the same plots as Figure 2.1, but with a logarithmic x-axis to more clearly show the low-training-data cases. Error bars were explicitly determined for the cases of 27, 162, 648, and 21546 images, and logarithmically interpolated for all other points. For each error bar to be determined, the model was retrained with the same amount of data four times, and the standard deviation of the four F1 scores was computed. For the two high-training-data cases, bootstrap resampling was used to approximate the effect of sampling from a larger underlying population. Because each error bar is computed from only four trials, the error of the error is itself high.

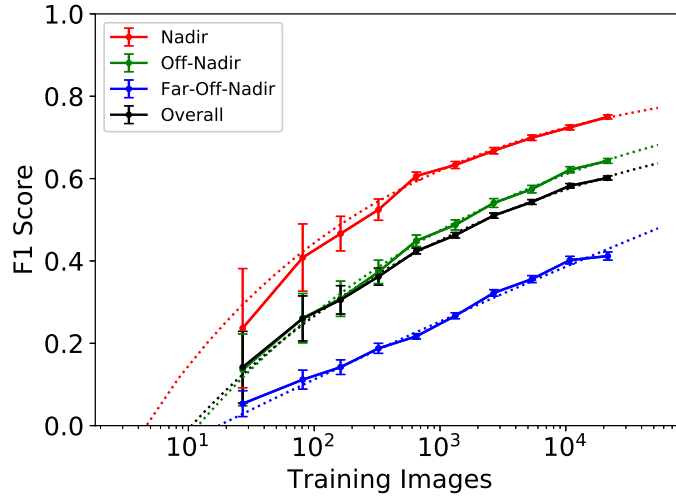


Figure 2.3: Identical to Figure 2.1, but with a logarithmic x-axis.

This careful analysis of errors, although computationally expensive, provides

further context for the results. In particular, it shows that models trained with abundant data are consistent in their performance, while models trained with little data can vary widely in performance. When training on 21,546 images covering 1064 locations, the overall F1 score varies by only a few hundredths. Training on 27 images of a single location, however, produces F1 scores ranging from near zero to more than 0.2 depending on the location.

2.3 Curve Fitting and Extrapolation

It is useful to fit curves to the results in Figure 2.3, both for better understanding of the results and to enable extrapolation to other amounts of training data. Several functional forms were tried, including a logarithmic curve (which would make a straight line on Figure 2.3 because of the logarithmic x-axis) as well as a constant minus an exponentially-decaying term. However, a much better concordance with the data is achieved with a constant minus an inverse power law term, the form used to generate the fits actually shown in Figure 2.3 (and Figure 2.1). We’ve already seen this expression — it’s the same function that’s been shown to be a good fit for learning curves in classification problems.

$$y = a - \frac{b}{x^c}$$

Figure 2.4: A constant minus an inverse power law. The parameters a , b , and c are all positive. Here, the input x is the number of training images, and the output y is the predicted F1 score.

With fitted functions in hand, we can make some extrapolations. If this simple fit holds to arbitrarily large training data set sizes, it would imply the existence of an asymptotic maximum F1 score as the amount of data is increased. The maximum (if infinite data existed) would be 0.87 ± 0.04 for nadir and a statistically equal 0.90 ± 0.06 for off-nadir. (The far-off-nadir value cannot be measured to reasonable accuracy without training many more models to reduce statistical uncertainty.) It should be emphasized that these scores are significantly higher than what was actually observed, and it is an open question whether such an extreme extrapolation is valid.

Even if the simple function doesn’t hold to infinitely large amounts of training data, we stand on firmer ground in asking what would happen if we’d simply had twice as much data as was actually available. The answer there is straightforward: all the effort of doubling the data set size is predicted to produce only a modest 3% improvement in overall F1 score. The relative slopes of the curves indicate that the far-off-nadir score would see the greatest benefit from additional data, with off-nadir and nadir showing smaller gains.

As a technical aside, we can check whether the fluctuations of the points in Figure 2.3 about their respective fitted curves are consistent with the given error bars on those points. We numerically calculate the effective degrees of freedom for each regression and assume the error-bar-normalized residuals follow the corresponding χ^2 distribution. The resulting p-values, which are respectively .43, .23, .97, and .31 for nadir, off-nadir, far-off-nadir, and overall, are quite plausible for a correct fit.

2.4 Putting the Method to the Test

Suppose that instead of having the full training data set, we only had one-sixteenth of it. With only 1,323 images to train on instead of 21,546, we want to estimate the improvement from acquiring sixteen times more training data. Figure 2.5 shows what happens when we extrapolate as above. Here we've simply ignored the points with more than 1,323 images when fitting the curve. This test is somewhat artificial in that the low-training-data samples were not restricted to all be drawn from the same one-sixteenth of the full data set. However, that is not expected to materially change the result.

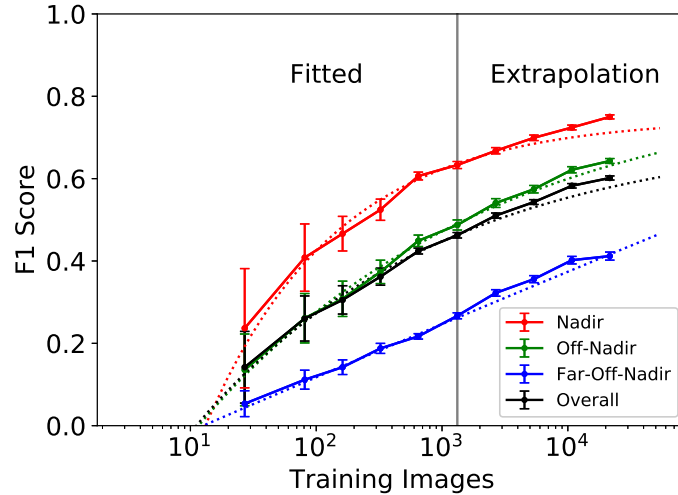


Figure 2.5: F1 score vs. amount of training data (solid lines), along with fitted curves (dotted lines) that are based on only the points with 1,323 training images or fewer.

From starting with 1,323 images, the table in Figure 2.6 shows the actual and predicted performance increases from doubling the training data and from increasing it by a factor of 16. In each case the prediction was correct to within a factor of two, and in most cases it was correct to within 25%. These figures

refer to the estimated improvements in the F1 score; the corresponding percent errors in the estimated F1 scores themselves are much smaller. The measured and predicted results tend to gradually diverge as training data is increased. However, the F1 scores increase with more data, which constrains the growth of the percentage error. As a result, for this case study the method gives an estimate for a 16-fold increase in training data that’s about as good as its estimate for a mere 2-fold increase. For a point of comparison, fitting the data with a simple logarithmic curve, instead of the constant minus a power law, produces a maximum percent error that’s almost twice as large as seen here.

| | 2x Data | | | 16x Data | | |
|---------------|-----------|---------|------------|-----------|---------|------------|
| | Measure | Predict | % Error | Measure | Predict | % Error |
| Nadir | .035±.011 | .028 | -19.8±6.6% | .117±.010 | .075 | -35.5±3.0% |
| Off-Nadir | .053±.016 | .043 | -19.9±6.1% | .155±.014 | .141 | -9.3±0.8% |
| Far-Off-Nadir | .056±.011 | .039 | -29.8±5.7% | .145±.012 | .154 | +6.1±0.5% |
| Overall | .048±.009 | .036 | -24.9±4.8% | .139±.008 | .115 | -17.2±1.0% |

Figure 2.6: Comparison of measured and estimated improvement with a 2-fold data increase and a 16-fold data increase. “Measure” is the actual increase in F1 score, “predict” is the increase predicted by the model, and “% Error” is the percentage error between them.

2.5 A Look at Ensembling

A standard technique to increase deep learning model performance is to replace a single neural net with an ensemble of neural nets, which can be helpful even if all the neural nets in the ensemble have the same architecture. We can ask whether the amount of training data influences the efficacy of this technique.

XD.XD’s original code²³ uses an ensemble, where each model in the ensemble is trained with a different quarter of the original data being held out for validation. That makes it difficult to isolate the improvement due specifically to ensembling. After all, if the ensemble outperforms one of its constituent models, it that because of the intrinsic benefits of the ensemble, or because the one model in isolation has access to 25% less data?

To measure the effect of ensembling directly, we instead train each model in the ensemble on the same training data set, withholding no data for validation (which we can forego here because appropriate training times were already found above). This was done in a “low-data” case with 216 images covering 8 locations, as well as a “high-data” case with all 28728 images covering 1064 locations.

In the low-data case, an ensemble of four models performs a full 10% better than the average performance of its constituent models. In the high-data case, however, the improvement is only 3%. This suggests that ensembling is more effective when training data is limited and provides less benefit when training

data is abundant.

2.6 Conclusions

We have shown that the same functional form used to fit learning curves is also a good fit to plots of F1 vs data set size for this specific combination of problem, model architecture, data set, and evaluation metric. We conjecture that it would be a good fit for many other segmentation problems using this metric as well.

The takeaways from this analysis are of two kinds: general methods for studying training data set size dependence, and specific results for building footprint detection.

As for general methods, the key takeaway is that being thoughtful about how much training data one needs is an important part of any well-designed deep learning project. The approach of fixing model architecture and varying training data can provide insight into what can, and can't, be gained from having more data. Although the needs of each project differ, aiming for the “sweet spot” with enough data to ensure consistent model performance but not so much data as to pay a high price for diminishing returns, is ideal. The methods for error estimation and curve fitting applied here are helpful tools for finding that sweet spot. With extrapolation, it's possible to begin gaining insight even before all the data is in.

For building footprints specifically, the key takeaway is the high return from even limited training data. It doesn't take “millions of images” to train a deep learning geospatial model — fewer than a thousand quarter-of-a-square-kilometer tiles, containing 27 views of the same few square kilometers of ground, are enough to get two-thirds of the performance of a training data set that's more than thirty times larger. The same feature of F1-vs-data-amount curves that creates diminishing returns from increasing the training data also makes model performance surprisingly robust when the amount of data is low.

Still, questions remain about how generalizable all of this is, and whether it carries over to other geographic regions or model architectures. Those questions will be taken up in subsequent chapters.

Chapter 3

Multiple Geographic Locations

When training a deep neural network to identify building footprints in satellite imagery, having more training data never hurts. But how much does more data help, and when is it worth the cost and difficulty of procuring it? We’ve seen that performance rises rapidly with training data quantity when data is limited, but huge amounts of data give diminishing returns. In this chapter, we will explore how *geography* affects that trend, by studying performance versus training data set size with satellite images of four cities from around the world. We’ll see that the overall pattern of diminishing returns holds across the board, but that there are also some city-specific characteristics. In the next chapter, we’ll look more closely at these city-specific features and see what happens when you train a model on one city and test it on another.

This chapter continues the look at robustness of model performance with limited training data, as begun in Chapters 1 and 2. However, those chapters are not prerequisites for what follows.

3.1 Method

The SpaceNet 2 Challenge²⁴ saw the release of electro-optical imagery of four world cities from the WorldView-3 satellite, along with accompanying building footprint labels. We’ll use that data here, allowing for a comparison of the diverse urban environments of Las Vegas, USA; Paris, France; Shanghai, China; and Khartoum, Sudan (Figure 3.1).

The SpaceNet 2 imagery is chipped into tiles of 650 pixels on a side. At a ground sample distance of around 0.3m, that corresponds to about 200m. Weighting the four cities equally, the average tile has 21 buildings.



Figure 3.1: Scenic views of (clockwise from top left) Vegas, Paris, Shanghai, and Khartoum.

The deep learning model architecture used here for identifying building footprints is a lightly-modified version of a prizewinning model architecture²⁵ submitted to the SpaceNet 4 Challenge by user “XD_XD.” The modified version, which was also used in the previous study of Atlanta (Chapters 1 and 2), uses one neural net instead of an ensemble.

For each city, models are trained with four different quantities of training data. These quantities — 12, 48, 192, and 759 tiles — are separated by multiplicative factors of about four. In addition, a fifth set of models is trained using tiles from all four cities at once. These combined models use the same number of tiles *per city* as the individual city models, so the combined model with the most data is trained on 3036 tiles, or 759 per city. Most one-city models are trained for approximately 100,000 image impressions and most combined models are trained for approximately 400,000 image impressions. After training, every model is tested against four test datasets, one for each city. This study uses the same evaluation metric²⁶ as SpaceNet building footprint challenges: an F1 score describing how often predicted building footprints are similar to ground truth building footprints.

For each of the twenty combinations of geography/training data amount, the training and testing procedure is repeated four times with a different randomly-selected subset of the data each time. The reported F1 score is the average of the four trials, and its error bar is half their standard deviation. (It's a half because the error of a mean falls as the square root of the number of samples.) This method represents an improvement over the technique used in the previously-described Atlanta study. The averaging makes the data less jittery and the error bars smaller. Having a more precise measure of the average model performance allows us to perceive small geographic differences that might otherwise get lost in the random performance fluctuations that occur each time a model is trained anew.

3.2 Results and Conclusions

The main results from this campaign of training and testing 80 models can be found in Figure 3.2. The figure shows model performance, measured by average F1 score, versus the amount of training data used. The solid lines with error bars are the actual results, and the accompanying dotted lines are curves fitted to those results. The color of each graph indicates the city where performance is being evaluated (as specified in the figure's legend). For each city, two scenarios are shown. The lower graph, shown in a darker shade, is the performance of a model trained only on data from that city itself. The higher graph, shown in a lighter shade, is the performance of the "combined model" trained on data from all four cities. Note that the x-axis is the amount of training data per city, so the combined-model results are based on four times more training data than the individual city results alongside them.

There are a lot of conclusions to be drawn from this, so let's give it a closer look!

To start, the plot exhibits the trend noted at the start of this chapter: When data is limited, performance rises rapidly with increased data, but that rate of growth cannot possibly be sustained once data becomes abundant. The logarithmic x-axis of Figure 3.2 visually disguises just how extreme the change is. To take the results trained on only Paris, for example, the slope of a line connecting the two lowest data points is sixty times greater than the slope of a line connecting the two highest ones. The trend can be seen more clearly in Figure 3.3, which shows the same data with a linear x-axis.

This has a fortunate consequence. Compared to labeling a large portion of a city or of a collect, one can get most of the performance with a small fraction of the training data. For all eight scenarios shown in Figure 3.2, training on 48 tiles per city instead of 759 gives more than 3/4 of the performance using a mere 1/16 of the data. The total area of 48 tiles is less than two square kilometers.

The dotted lines in Figure 3.2 are empirical fits to the data points, and here we see another result from our previous analysis of Atlanta that holds up across a

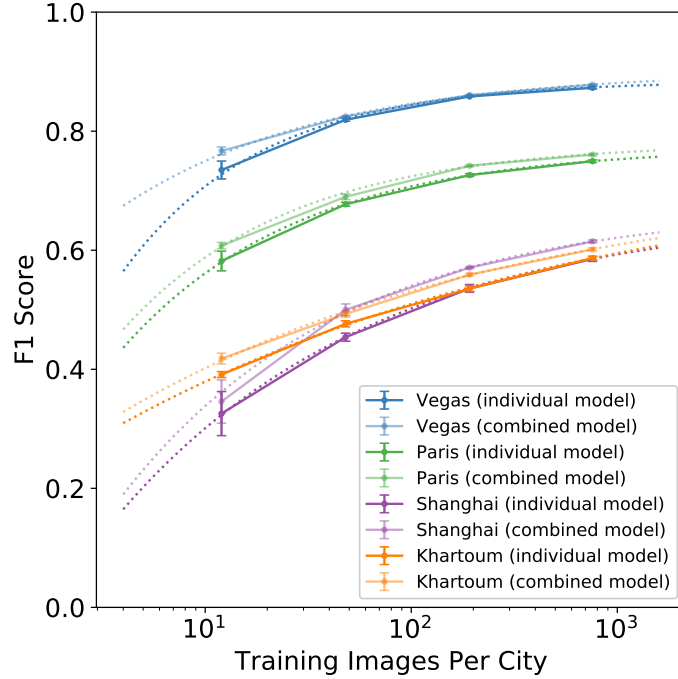


Figure 3.2: Average F1 score versus number of training images per city. “Individual model” denotes models trained only on the city for which their F1 score is shown, while “combined model” denotes models trained on all four cities (i.e., with four times as much training data).

variety of geographies. These curves follow a “learning curve” functional form: a constant minus an inverse power law term. Having a simple empirical fit like this is useful for interpolation and extrapolation, as shown with the Atlanta data set in Chapter 2. At first glance, the curves might seem *too* good, in that the deviations between the points and the fitted curves are small compared to the error bars. However, that’s not a surprise when fitting four points using a curve with three free parameters, and the χ^2 p-values calculated with the effective degrees of freedom are consistent with an appropriate fit.

Although the different scenarios can all be fit with the same type of function, clearly they are not the same. The most noticeable difference is the wide range of F1 scores among cities given the same amount of training data. When it comes to identifying building footprints, some cities are more challenging than others.

Among these training scenarios, there’s a more subtle trend that also has implications for a real-world use case. Suppose I have a fixed amount of data for

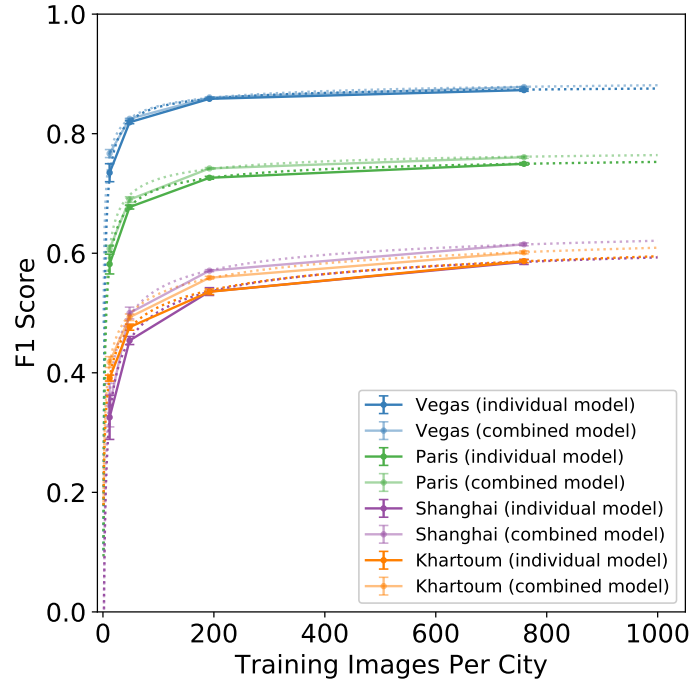


Figure 3.3: Identical to Figure 3.2, but with a linear x-axis.

each city, and I want to get the highest performance possible. Should I train a separate bespoke model for each city, or should I pool all the data together to train one general-purpose model? In other words, does the benefit from having four times as much data outweigh the challenge of trying to create one generalized model for four cities from around the world? As Figure 3.2 shows, a model trained on four cities' worth of data performs no worse than a model trained on the quarter of that data from the specific city where the model is being tested. That means the model architecture's parameter space is big enough to hold four cities' worth of learning, so it does not force any trade-off between performance and generality. In fact, it's better than that: the model trained on the larger combined data set of all four cities consistently gets slightly *higher* performance than the city-specific models.

Figure 3.4 gives an example of reading Figure 3.2 to see this effect in action. If my only goal is to make the best possible model for, say, Paris, and all I have are 12 imagery tiles of Paris and 12 tiles apiece for three other cities, then a model trained on all 48 tiles from the four cities will, on average, outperform a city-specific model trained on the 12 tiles from Paris alone. Of course, if I can increase my Paris training data from 12 to 48 tiles, that will bring about

an even greater benefit to the Paris model. But the key conclusion is that increasing training data by incorporating different geographies not only makes a more generalized model—it also makes a model that in every case outperforms city-specific models trained on less data.

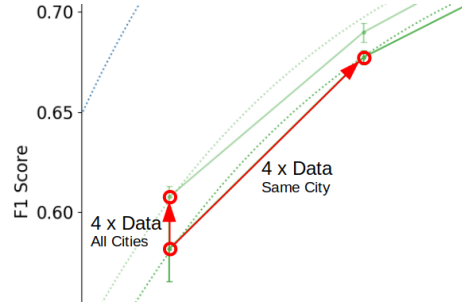


Figure 3.4: Detail of Figure 3.2, showing the change in Paris F1 score from quadrupling the amount of training data. Quadrupling the training data by getting four times more data from the same city (diagonal line) shows the larger improvement, but quadrupling the training data by incorporating equal amounts of data from three other cities (vertical line) still helps some.

Finally, since we’ve been comparing the results of this analysis to a previous analysis of Atlanta, it’s worth seeing the results side by side, as shown in Figure 3.5. The Atlanta data shows lower and more steeply rising F1 scores for the same number of training images. The cause is not a geographic difference, but rather a difference in how the data sets are structured. Although the Atlanta tiles are larger ($\sim 450\text{m}$ on a side instead of $\sim 200\text{m}$), they are lower-resolution and also geographically redundant: each Atlanta location is shown in 27 different tiles from different angles. As a result, five times more Atlanta tiles are needed to cover the same amount of area. This, along with the intrinsic difficulty of lower-resolution and off-nadir imagery, affects the shape of the curves and increases the number of tiles needed to achieve performance comparable to the SpaceNet 2 cities.

This discussion of training models on different geographic areas has focused on overall trends. These include the utility of small amounts of data and a simple empirical fitting function. The value of combining training data to build a global model has also been shown. In the next chapter, we’ll take a closer look at two specific questions about the variations between cities: First, what happens if we train a model on one city and test it on a completely different city? Second, why does the performance seem to differ among cities when using the very lowest amounts of data?

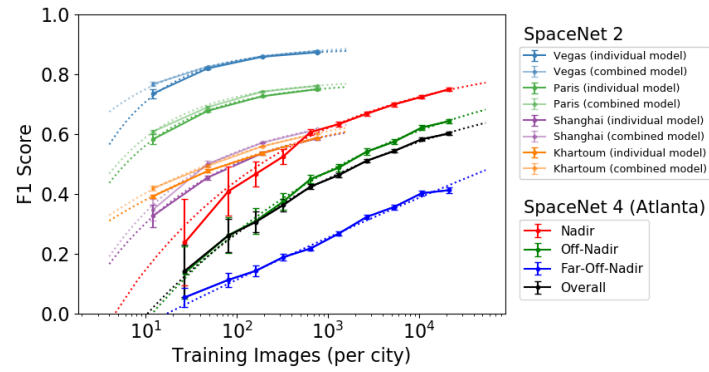


Figure 3.5: F1 score versus amount of training data for the SpaceNet 4 Off-Nadir Atlanta data set and the SpaceNet 2 data set of world cities.

Chapter 4

Exploring Geographic Differences

When it comes to the relationship between geospatial neural network performance and the amount of training data, do geographic differences matter? In the previous chapter, we examined this question by training the same building footprint model using various amounts of data from four different cities: Las Vegas, Paris, Shanghai, and Khartoum. That led to a plot (Figure 4.1) of performance for each city, either using a model trained on the city in question or using a model trained on the combined data of all four cities. In this chapter, we'll take a closer look at two questions that went unanswered in the previous chapter. First, what happens if we take a model trained on one city and apply it to a different one it's never seen before? And second, why does just one of the cities — Khartoum, Sudan — respond to a training data deficit more resiliently than the others?

4.1 Transferability

To assess model transferability, models trained on one city are tested on the others. The training is repeated four times for each training city, using 759 randomly selected tiles each time. Figure 4.2 shows the resulting performance, as measured by average F1 score.

Not surprisingly, the best results are achieved when the model is tested on the same city it's been trained on. Looking at the off-diagonal terms, none exceeds those on the diagonal. Beyond that, the matrix is strongly asymmetric in places. For example, a model trained on Khartoum and tested on Vegas does far better ($F1=.44$) than one trained on Vegas and tested on Khartoum ($F1=.07$). This illustrates that transferability is not commutative.

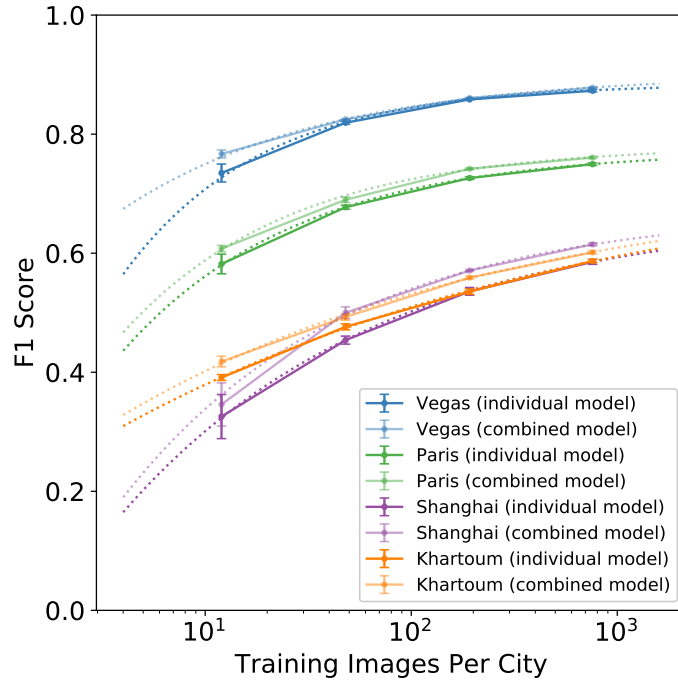


Figure 4.1: Average F1 score versus number of training images per city. “Individual model” denotes models trained only on the city for which their F1 score is shown, while “combined model” denotes models trained on all four cities (i.e., with four times as much training data). Identical to Figure 3.2.

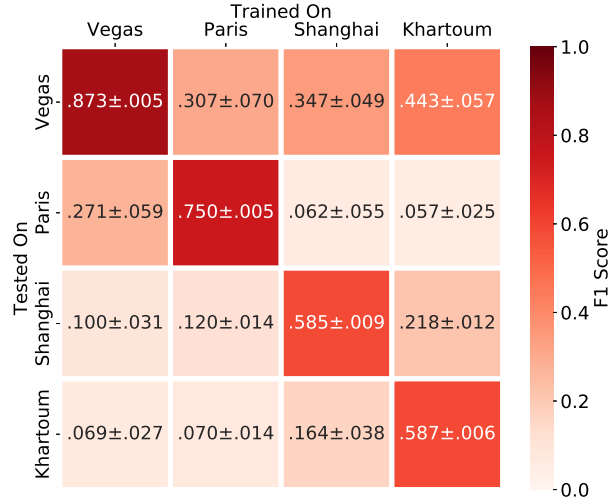


Figure 4.2: Mean F1 score for models trained on one city (horizontal axis) and tested on another (vertical axis). Training is done with 759 tiles that are 200m on a side.

We can also use Figure 4.2 as a way of understanding which cities are most similar in the appearance of their imagery, at least as concerns traits relevant to building footprint identification. For each pair of cities we assign a similarity score as defined in Figure 4.3.

$$S_{A,B} = \sqrt{\frac{F_{A \rightarrow B} F_{B \rightarrow A}}{F_{A \rightarrow A} F_{B \rightarrow B}}}$$

Figure 4.3: Similarity score for two cities, denoted A and B . S is similarity, F is an F1 score, and the subscript $A \rightarrow B$ means the model is trained on city A and tested on city B .

Then, we can make a “map” by plotting the network of cities in such a way that the more dissimilar cities are pushed further apart, as shown in Figure 4.4.

Khartoum and Paris are the two most disparate cities; Paris is the most unique overall. An important point is that this might be due to meaningful features on the ground (e.g., Paris’s abundance of trees or distinctive architecture), but it could just as easily be due to incidental details of this collect (e.g., lower light

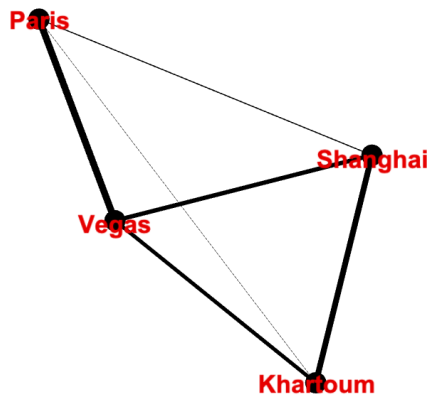


Figure 4.4: Graph of cities, with proximity based on similarity score.

in the Paris photo, or the choice to include a larger amount of rural terrain in the area of interest). Using more cities could help elucidate which factors are actually important.

4.2 Geography and the Low-Data Falloff

We'll now switch gears and consider an anomalous feature back in Figure 4.1. As the amount of training data is decreased, model performance declines along with it. By the time there's very little data, on the order of 12 tiles per city, performance plunges with even a small reduction in data. Three of the cities seem to fall off almost in unison, but the results for Khartoum in Figure 4.1 show a somewhat more gradual decline in performance as training data is reduced. This qualitative observation can be seen quantitatively in the numeric parameters of the fitted curves. Those curves (shown as dotted lines in Figure 4.1) follow a constant minus an inverse power law term. The power law exponents for the six curves from testing on cities other than Khartoum vary from city to city, but their average is $0.48 \pm .10$ (taking their standard deviation to be the uncertainty). The two curves from testing on Khartoum, however, have an average exponential term of only $0.20 \pm .02$, reflecting a less-rapid change in performance in the low-data regime. Additionally, the uncertainty for the model trained on Khartoum with 12 tiles is notably lower than the other individual cities' 12-tile models.

In short, Khartoum is more resilient in a data-constrained situation. And since Khartoum is neither the best- nor worst-performing city in this regime, the issue appears not to be a simple matter of overall difficulty. Instead, something specific to Khartoum is going on.

So why is Khartoum different? We can rule out a chance statistical fluctuation.

Here, we are rewarded for our patience in running enough trials to generate error bars. Comparing the one-standard-deviation error bars on the curves for Khartoum and Shanghai in Figure 4.1 shows that it is quite unlikely that their diverging behavior at low training data levels is due to chance alone.

We can also rule out an unusual distribution of building sizes in Khartoum. Figure 4.5 shows that the city’s distribution of building sizes is not particularly different from the other cities. Even the bimodal peak in Khartoum’s building size distribution is not unique, as Las Vegas shows the same feature.

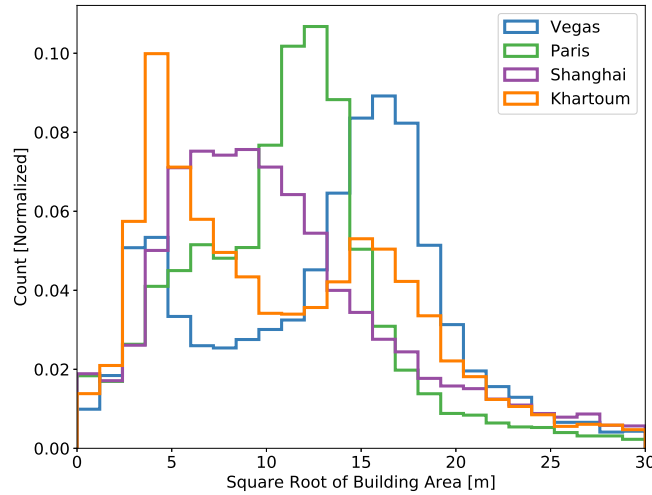


Figure 4.5: Normalized histogram of building sizes in each city, as a function of the square root of their area.

Khartoum ranks neither first nor last in median building size, average count of buildings per tile, or percentage of land covered by buildings. As for the collect, the Khartoum imagery is not unusually dark or bright. It is the most off-nadir, but only barely so (compare Khartoum’s 25.7 degrees off nadir to Shanghai’s 20.5, both too low to see the largest effects²⁷ of off-nadir angles).

What, then, can explain the difference? A visual inspection shows just how different Khartoum looks compared to the other cities. Figure 4.6 shows typical tiles from Khartoum and Vegas. We’ll compare these two cities in terms of low-level features likely to play a role in the first layers of the neural net: colors and edges.

Figure 4.7 shows the range of colors in eight randomly selected tiles from Vegas and Khartoum. The colors are represented in the HSV color space,²⁸ with hue (color of the rainbow) on the x-axis and value (darkness) on the y-axis. The

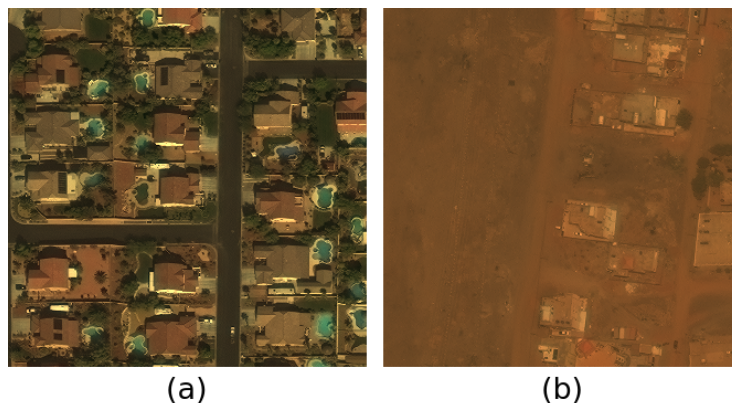


Figure 4.6: Typical tiles from (a) Vegas and (b) Khartoum.

saturation (intensity) of each point indicates how frequently pixels with the given combination of hue and value occur in the imagery. Vegas is a colorful place, with recognizable features such as the distinctive light green of backyard swimming pools. In contrast, the color palette of Khartoum is more restrained. (Another look at the role of color using this same data set can be found in a [DownLinQ](#) post.²⁹)

As for edges, we can get a feel for the distribution of edges in the sample tiles by applying a simple edge detector with image editing software. Figure 4.8 shows the result. Most edges in this Khartoum tile are associated with buildings, whereas edges are prevalent in both building and non-building parts of the Vegas image. That’s due to Khartoum having less variation in color and also due to other aspects of the scenery, such as the lower variation in ground cover type in Khartoum.

All of this suggests a hypothesis for why building footprint identification holds up better with little training data in Khartoum. It may be that a rule of thumb that “edges imply buildings” works exceptionally well in Khartoum, and such a rule would require very little training data to learn. The high variety of scenery in a city like Vegas (for which the high variety of colors and edges serve as simplistic proxies) would permit no such rule for low training data. Such variation, despite being a liability with low training data, nevertheless becomes advantageous given enough training data for the model to learn it.

While further evidence for this hypothesis could possibly come from hand-engineering an edge-based building footprint detector, actually proving it is a more difficult matter. One route would be to generate synthetic data, building synthetic city views with mixes of attributes to find out which ones lead to reduced performance degradation at the lowest training data levels. Another route would be to investigate the actual features being learned by the models with an explainable AI approach. Short of that, an understanding of this issue,

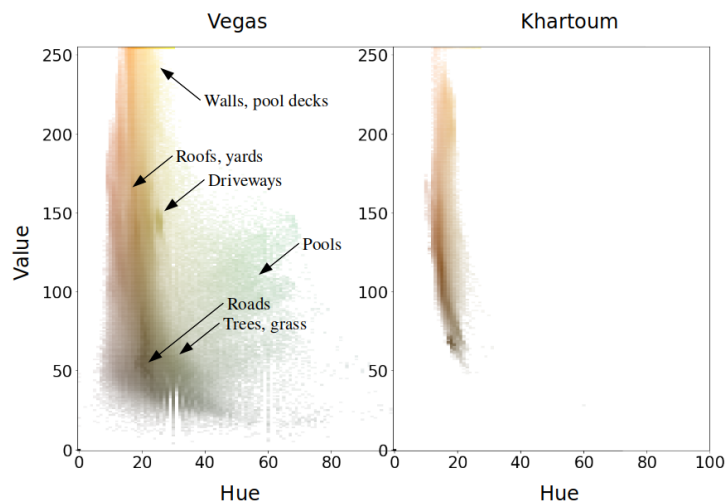


Figure 4.7: Distribution of colors across eight randomly-selected tiles from each city. The saturation of each pixel in the histogram scales logarithmically with how often the corresponding hue+value combination occurs in the tiles. Hues from 100 to 180 have been left off because the imagery includes almost no pixels in this range. Example objects of different colors are indicated in the Vegas plot, but this list is not exhaustive.

like the issue of transferability discussed above, would benefit from studying a greater number of cities.

4.3 Next Step

In this chapter, we’ve investigated model performance in two especially challenging scenarios. First, we looked at models trained on a different city from where they were being put to the test. The performance matrix was shown not to be symmetric, and a way to visualize similarity was proposed. Second, we looked at models trained on very little data (12 tiles = half a square kilometer) to understand why some face steeper performance declines than others in the low-training-data limit. Comparing Vegas and Khartoum, the latter shows lower variation in hue, as well as a stronger relationship between edges and buildings. This inquiry demonstrated the challenges of uncovering what’s happening under the hood in deep learning.

In the next chapter, we’ll look at the effects of changing a different variable. Instead of changing geographic location, it’s time to head back to Atlanta and try changing the model architecture itself.

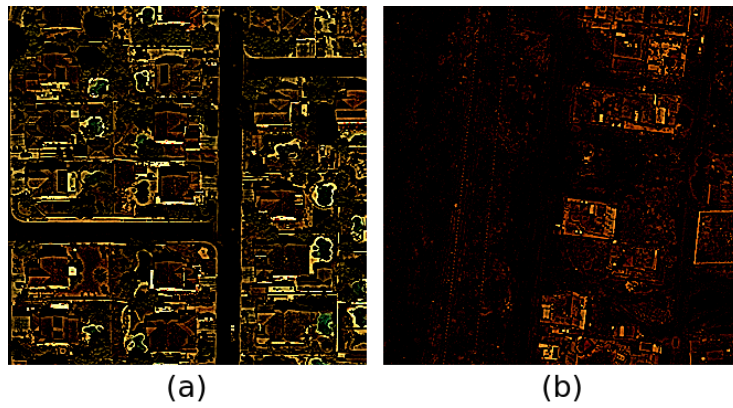


Figure 4.8: Edges in the tiles of (a) Vegas and (b) Khartoum shown in Figure 4.6. Images are processed with a difference-of-gaussians filter followed by an increase in brightness and contrast.

Chapter 5

Architectures Compared, & Final Thoughts

Deep learning models for interpreting satellite imagery show increasing performance as the amount of training data is increased. We’ve looked at the details of how this plays out, but our previous studies of this topic have suffered from a weakness: they’ve all used just one model architecture, meaning the layout of artificial neurons and other details of the algorithm were the same each time. In this chapter, we’ll recreate our first analysis with a whole new model architecture, to see what changes and what stays the same.

This is the last chapter in this monograph, so we’ll close with some overall conclusions. The previous chapters are not prerequisites for what follows.

5.1 Two Model Architectures

To explore the dependence of model performance on the amount of training data, we train a model to identify building footprints in satellite imagery. Our imagery comes from the SpaceNet 4 data set,³⁰ which contains 27 views of Atlanta taken with Maxar’s WorldView-2 satellite. The imagery, with a ground sample distance of about half a meter, is chipped into tiles of about 450m on a side. The model architecture is a lightly-modified version of the 5th-place SpaceNet 4 prize winning model³¹ submitted by challenge participant XD_XD. This model features fast inference, and the version used here replaces the original ensemble of three neural networks with a single neural network to speed up training. For brevity, we will refer to this model architecture as “architecture A.” Performance is evaluated with an object-wise F1 score (the SpaceNet metric³²) for approximately-correct building footprints. Figure 5.1 shows how that performance varies with the amount of training data. The red,

green, and blue lines show the results broken down by viewing angle, while the black line is an overall result. Details on this can be found in Chapters 1 and 2.

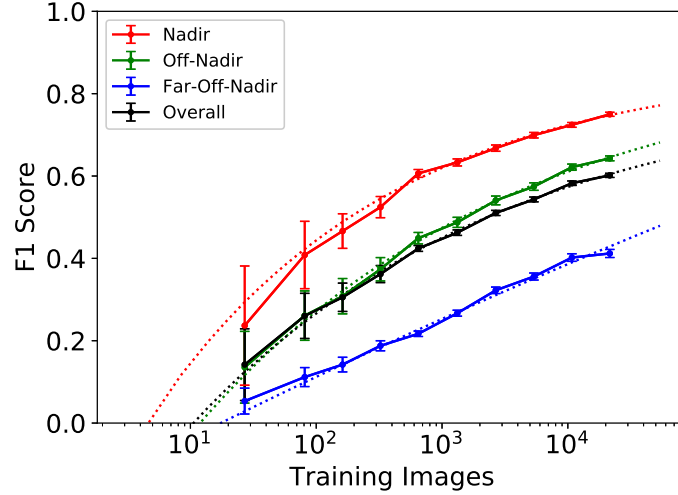


Figure 5.1: Performance of model architecture A, as measured by F1 score, versus number of images used for training. Dotted lines are fitted curves.

Next, we repeat this process with a different model architecture to see if the results in Figure 5.1 hold up. Our second architecture is a lightly-modified version of the 1st-place SpaceNet 4 submission,³³ from challenge participant cannab. The original submission’s ensemble of 28 neural nets is pared down to an ensemble of four for faster training. Each neural net uses a U-net layout with a SE-ResNeXt50 encoder for pixel segmentation. Possible positives are subjected to further winnowing prior to generating vector building footprints. We’ll refer to this model architecture as “architecture B.”

Architectures A and B both use U-nets to create pixel maps from which building footprint polygons are generated, but there the similarities end. The two architectures use different encoders and different amounts of post-processing. Architecture B is an ensemble while A is not. Two model architectures clearly can’t span the space of everything that’s possible, but these are different enough to be a reasonable test of any highly-model-specific tendencies.

With that in mind, we measure performance as a function of training data quantity for model architecture B. We follow the procedure in Chapter 1, except for using the method to calculate error bars introduced in Chapter 3. Figure 5.2 shows the result.

The results of model architectures A and B are broadly similar. In both cases,

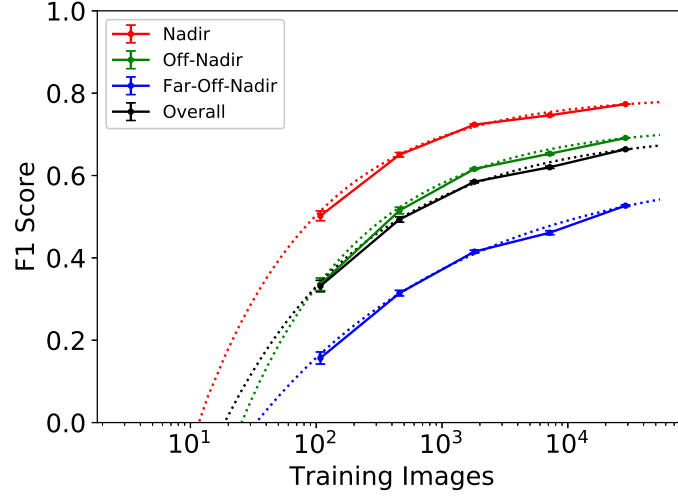


Figure 5.2: Performance of model architecture B, as measured by F1 score, versus number of images used for training. Dotted lines are fitted curves.

a rapid performance increase with low amounts of training data gives way to diminishing returns with higher amounts of data. Performance initially rises more quickly with architecture B, getting closer to its maximum value with less training data. Architecture B also shows more consistent performance, with shorter error bars (even after taking into account a factor of two from a change in the analysis procedure).

As previously shown for architecture A, architecture B results can be roughly fit by curves scaling as a constant minus an inverse power law. Such fitted curves for architecture B are included in Figure 5.2. In this case, the actual results do deviate slightly from the simple model, with the curves underpredicting the measured results at 67 tiles and overpredicting the results at 266 tiles. This deviation is too big to ascribe to a chance statistical fluctuation. In fact, a χ^2 test rejects the hypothesis that the simple curves alone can fully explain the variation in the observed performance. Nevertheless, the deviations from the fitted curves are still small compared to the dynamic range of the performance, so the model is still useful even though it is something of an oversimplification.

Finally, we ask what seems like a simple question: which architecture is better? Figure 5.3 shows the architecture B results again, which the architecture A results overlaid in lighter colors.

At around 1000 training images, architecture B shows much higher performance than architecture A. By around 20,000 training images, architecture B only

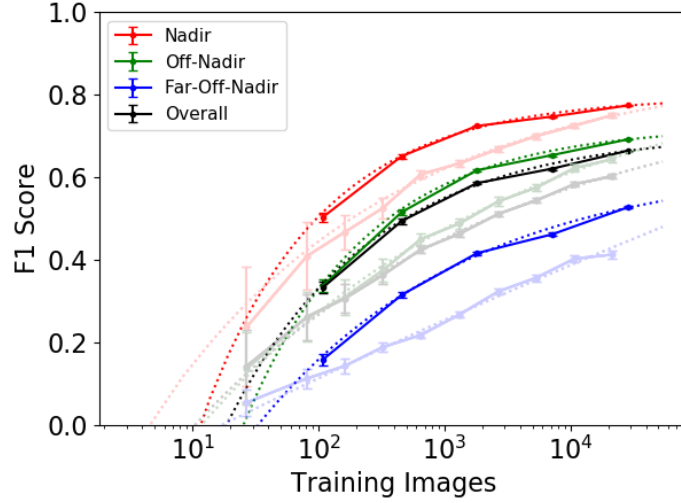


Figure 5.3: F1 score versus number of training images for architecture A (lighter shades, from Figure 5.1) and architecture B (darker shades, from Figure 5.2).

barely outperforms A. Extrapolating with the fitted curves indicates that, with enough data, architecture A overtakes architecture B.

What this demonstrates is that the model architectures cannot be absolutely ranked. It’s not always possible to answer the question “which architecture gets better performance?” — unless one specifies how much training data is to be used. Deep learning papers often pronounce a model to be a success if it outperforms a previous best model on some canonical data set. But a canonical data set necessarily has a fixed size. The results with architectures A and B give a clean example of a case where a comparison using a single quantity of training data cannot tell the full story.

5.2 Project Conclusions

To close this monograph on the robustness of models with limited training data, we revisit some of the key points we’ve come across along the way.

- Foundational mapping model performance rises with data rapidly when data is scarce and slowly when data is abundant. This holds across model architectures and across geographies. Having a whole city’s worth of data is not a requirement for geospatial deep learning. A large fraction of that performance can be achieved with a small fraction of the data.
- Various statistical methods enable more thorough analyses. Error bars

are critically important for deep learning research, revealing when results are repeatable and when they're even real.

- To get the maximum performance across multiple cities with a fixed (and equal) amount of data for each one, building a specific model for each city gives worse results than piling all that data together to train one general-use model.
- A simple functional form works well for fitting building footprint model performance across a variety of architectures and geographies. Differences in the fitted parameters point to poorly-understood differences in the cities themselves.
- A deep learning architecture that performs the best with low training data might not be the best given lots of training data.

But in the end, the most important lesson is this: Geospatial deep learning works surprisingly well with surprisingly little training data. Millions of images are *not* required — the training data barrier to entry for geospatial deep learning is lower than many people think it is.

Links

- ¹<https://medium.com/the-downlinq/robustness-of-limited-training-data-for-building-footprint-identification-part-1-8c55810b5ef9>
- ²<https://medium.com/the-downlinq/robustness-of-limited-training-data-part-2-f51eb783823f>
- ³<https://medium.com/the-downlinq/robustness-of-limited-training-data-part-3-9df24c58c2>
- ⁴<https://medium.com/the-downlinq/robustness-of-limited-training-data-part-4-3b9fb5033abc>
- ⁵<https://medium.com/the-downlinq/robustness-of-limited-training-data-part-5-5a0b0518add9>
- ⁶<https://towardsdatascience.com/predicting-the-effect-of-more-training-data-by-using-less-c3dde2f9ae48>
- ⁷<https://medium.com/the-downlinq/the-satellite-utility-manifold-object-detection-accuracy-as-a-function-of-image-resolution-ebb982310e8c>
- ⁸<https://medium.com/the-downlinq/the-satellite-utility-manifold-object-detection-accuracy-as-a-function-of-image-resolution-ebb982310e8c>
- ⁹<https://medium.com/the-downlinq/panchromatic-to-multispectral-object-detection-performance-as-a-function-of-imaging-bands-51ecaaa3dc56>
- ¹⁰<https://medium.com/the-downlinq/the-good-and-the-bad-in-the-spacenet-off-nadir-building-footprint-extraction-challenge-4c3a96ee9c72>
- ¹¹<https://journals.aps.org/pra/abstract/10.1103/PhysRevA.45.6056>
- ¹²<http://papers.nips.cc/paper/803-learning-curves-asymptotic-values-and-rate-of-convergence.pdf>
- ¹³<https://arxiv.org/abs/1511.06348>
- ¹⁴<https://medium.com/the-downlinq/quantifying-the-effects-of-resolution-on-image-classification-accuracy-7d657aca7701>
- ¹⁵<https://spacenetchallenge.github.io/>
- ¹⁶<https://medium.com/the-downlinq/introducing-the-spacenet-off-nadir-imagery-and-buildings-dataset-e4a3c1cb4ce3>
- ¹⁷<https://medium.com/the-downlinq/the-good-and-the-bad-in-the-spacenet-off-nadir-building-footprint-extraction-challenge-4c3a96ee9c72>
- ¹⁸https://github.com/SpaceNetChallenge/SpaceNet_Off_Nadir_Solutions/tree/master/XD_XD
- ¹⁹<https://medium.com/the-downlinq/the-spacenet-metric-612183cc2ddb>
- ²⁰https://en.wikipedia.org/wiki/F1_score
- ²¹https://en.wikipedia.org/wiki/Jaccard_index
- ²²<https://medium.com/the-downlinq/introducing-the-spacenet-off-nadir-imagery-and-buildings-dataset-e4a3c1cb4ce3>
- ²³https://github.com/SpaceNetChallenge/SpaceNet_Off_Nadir_Solutions/tree/master/XD_XD
- ²⁴<https://spacenet.ai/spacenet-buildings-dataset-v2/>
- ²⁵https://github.com/SpaceNetChallenge/SpaceNet_Off_Nadir_Solutions/tree/master/XD_XD

²⁶<https://medium.com/the-downlinq/the-spacenet-metric-612183cc2ddb>

²⁷<https://medium.com/the-downlinq/the-good-and-the-bad-in-the-spacenet-off-nadir-building-footprint-extraction-challenge-4c3a96ee9c72>

²⁸https://en.wikipedia.org/wiki/HSL_and_HSV

²⁹<https://medium.com/the-downlinq/panchromatic-to-multispectral-object-detection-performance-as-a-function-of-imaging-bands-51ecaaa3dc56>

³⁰<https://spacenet.ai/off-nadir-building-detection/>

³¹https://github.com/SpaceNetChallenge/SpaceNet_Off_Nadir_Solutions/tree/master/XD_XD

³²<https://medium.com/the-downlinq/the-spacenet-metric-612183cc2ddb>

³³https://github.com/SpaceNetChallenge/SpaceNet_Off_Nadir_Solutions/tree/master/cannab