MNRAS **000**, 1–6 (2018) Preprint 5 September 2018 Compiled using MNRAS LATEX style file v3.0

Predicting Galaxy Metallicity from Three-Color Images using Convolutional Neural Networks

John Wu[1⋆] and Steven Boada[1]

[1] *Physics and Astronomy Department, Rutgers University, Piscataway, NJ 08854-8019, USA*

Accepted XXX. Received YYY; in original form ZZZ

ABSTRACT

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

1 INTRODUCTION

Things we are going to want to talk about in the introduction: Morphology-metalicity relation? There is certainly a mass-metallicity relation. Basically, we want to say why we think we can do this at all? What is the primary science driver behind why we think this will work? Is it really just the fundamental plane? We aren't telling the thing about the mass or the distance to the objects. I'll add some words we might want to use Large-area sky surveys, both on-going and planned, are revolutionizing our understanding of galaxy evolution. The on going Dark Energy Survey (DES; **?**) and planned Large Synoptic Survey Telescope (LSST; **?**) will survey vast swaths of the sky and create samples of galaxies much larger than any previously know. Spectroscopic follow-up will be key to a deep understanding of the properties of these galaxies, and constrain something something. But as the dataset continues to grow individual follow-up becomes increasingly impractical. Therefore, large spectroscopic surveys are needed to more fully understand the observable-mass relation of clusters. In this work, we propose to use supervised ML, specifically convolution neural networks, to analyze pseudo-three color images to predict galaxy metallicity. This paper is organized as follows: Section 2 we briefly introduce convolutional neural networks, discuss selection of the network's hyperparameters and outline training the network. In Section 3, we describe the acquisition and cleaning of the SDSS data sample. We present the main results in Section 4 and discuss the results in the context of previous works in Section **??**. In Section 7, we summarize the key results and conclude. Unless otherwise noted, throughout this paper, we use a concordance cosmological model ($\Omega_\Lambda = 0.7$, $\Omega_m = 0.3$, and $H_0 = 70$ km s^{-1}Mpc^{-1}), assume a Chabrier initial mass function (**?**), and use AB magnitudes (**?**).

2 CONVOLUTIONAL NEURAL NETWORKS

In recent years, neural networks have been able to accomplish a large number of tasks in the field of machine learning (LeCun et al. 1989). Image classification and regression problems are most readily solved by use of convolutions in mutliple layers of the network (see, e.g., Krizhevsky et al. 2012). Convolution neural networks (CNNs, or convnets) efficiently learn spatial relations in images whose features are about the same sizes as the convolution filters (or kernels) which are to be learned through training. CNNs are considered *deep* when the number of convolutional layers is large; visualizing their filters reveals that increased depth permits the network to learn more and more abstract features (e.g., from Gabor filters, to geometric shapes, to faces; Zeiler & Fergus 2013).

The input layer is simply an image of 128×128 pixels with three channels (RGB). Dieleman et al. (Galaxy Zoo Kaggle competition) have already shown that CNNs are capable of classifying the morphologies of such images of galaxies from the Sloan Digital Sky Survey (SDSS) with the same accuracy as citizen scientists. Other people have been done other cools things (like use simulated images to select real galaxies, etc). blah blah blah. Define overfitting: a specific and usually not general set of features are learned and misapplied to the test set data.

We split our training sample of $\sim 130,000$ images into sets of $90,000$ for training, $20,000$ for validation, and $20,000$ for testing. Training images are seen once every epoch, although usually each epoch is split into a number of mini-batches which are learned in parallel. Mini-batches are usually small (256?) and potentially not representative of the full training sample – another technique used to prevent overfitting. Each mini-batch is fed forward through the input layers, where a random fraction of connections $p = 0.25, 0.50$ are removed between each linear layer (dropout, Hinton et al. 2012; see subsection below). When the feed-forward network reports a prediction, whether a single or set of quantities, then a loss/cost function is used to compute how

incorrect the prediction ($\hat{y}$) is from the true value y. We use the root mean squared error (RMSE $\equiv \sqrt{\langle|\hat{y} - y|^2\rangle}$) loss function, and seek to minimize it. We use gradient descent for each mini-batch to adjust each weight parameter, and each fractional contribution of loss is determined by the backpropagation algorithm (cite original, LeCun et al.). The backpropagation algorithm is simply the chain rule applied to finite derivatives, and skipped? for any non-linear layers. The gradient is multiplied by the *learning rate*; batch-normalization is also applied in addition to a momentum term which ensures that the gradient is itself only changing slowly with each mini-batch.

3 DATA AND TRAINING

3.1 SDSS images

Our sample of galaxies are selected from the NYU Value Added Catalog (VAC) cite, to have gas phase metallicities. The metallicity estimates (**?**) are derived from spectroscopic measurements and for the purposes of this work are assumed to be "the truth." In addition, we take the derived stellar mass for each galaxy. We supplement the data from the VAC with data from SDSS Data Release 13 (DR13; **?**) for each galaxy, the right ascension and declination and the magnitude in each of the five SDSS photometric bands (u, g, r, i, z), along with associated errors. We require that galaxies magnitudes are $10 < ugriz < 25$ mag. Galaxies should have colors $0 < u - r < 6$ why?, high confidence ($z_{err} < 0.01$) spectroscopic redshifts greater than 0.2, check the petro mags and why we are using those.

From this input catalog we create RGB images with the SDSS cutout service[1]. Images are scaled to be 128×128 pixels in size, corresponding to $15'' \times 15''$ on the sky. This results in the native $0\farcs396$ pixelscale of the SDSS being rescaled to $0\farcs296$ per pixel. We create 142,176 three color images which serve as the main data set for this work.

3.2 Training in practice

Our training methods near convergence after 10 epochs, although additional training continue to improve the loss. In total, the training requires 25-30 minutes on our GPU and uses under 2 GB of memory (depending on batch size). k-fold ensemble methods take k times as long to train. Prediction using TTA takes two minutes for our full test set of about 20,000 images, or approximately 6 milliseconds per image (or a little over 1 millisecond per image without TTA). Super-convergent training the network takes about 10 hours on a GPU but only yield marginal improvements over the ensemble methods (RMSE = 0.0828 for super-convergent versus 0.0835 for 5-fold).

```
lr = 1e-1
learn.fit(lr, 2)

lrs = [1e-3, 1e-2, 1e-1]
learn.unfreeze()
```

[1] http://skyserver.sdss.org/dr14/SkyserverWS/ImgCutout/getjpeg

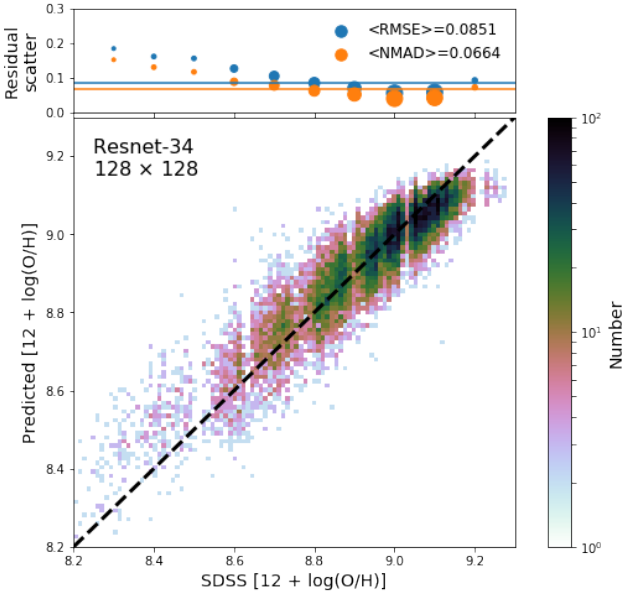

Figure 1. Our main result is shown in the main (lower) panel. We show metallicity predictions from *gri* imaging using a 34-layer residual convnet, compared to true metallicities from SDSS (Tremonti et al. 2004). In the upper panel, we show the residual, or difference between prediction and truth values, as a function of true metallicity. The size of markers is proportional to the number of galaxies in each true metallicity bin. The average RMSE and outlier-insensitive NMAD for the entire test set are 0.0851 and 0.0664 respectively, and are shown as horizontal colored lines.

```
learn.fit(lr, 1)
learn.fit(lr, 3, cycle_len=1, cycle_mult=2)
```

In comparison, Huertas-Company et al. 2015 train their network for 10 days on a GPU following the Galaxy Zoo architecture.

We evaluate predictions using not only the RMSE, which approaches the standard deviation for Gaussian-distributed data, but also the NMAD, or the normal median absolute deviation:

$$\mathrm{NMAD}(x) \approx 1.4826 \times \mathrm{median}\big(|x - \mathrm{median}(x)|\big), \quad (1)$$

where for a Gaussian-distributed x, the NMAD will also approximate the standard deviation, σ. NMAD is outlier-insensitive for non-Gaussian distributions and is useful for comparing scatter.

4 RESULTS

See figure for 50th percentile [O/H] predictions.

4.1 Diagnostics

Note by the way that our CNN trained on color images is able to predict better than by using only photometry (via random forest) and morphology (r-band CNN) independently. Perhaps this is due to the fact that morphology is correlated with stellar population age or something else. Color tells us the fundamental plane of SFR-M_*-metallicity but is biased by age and dust.

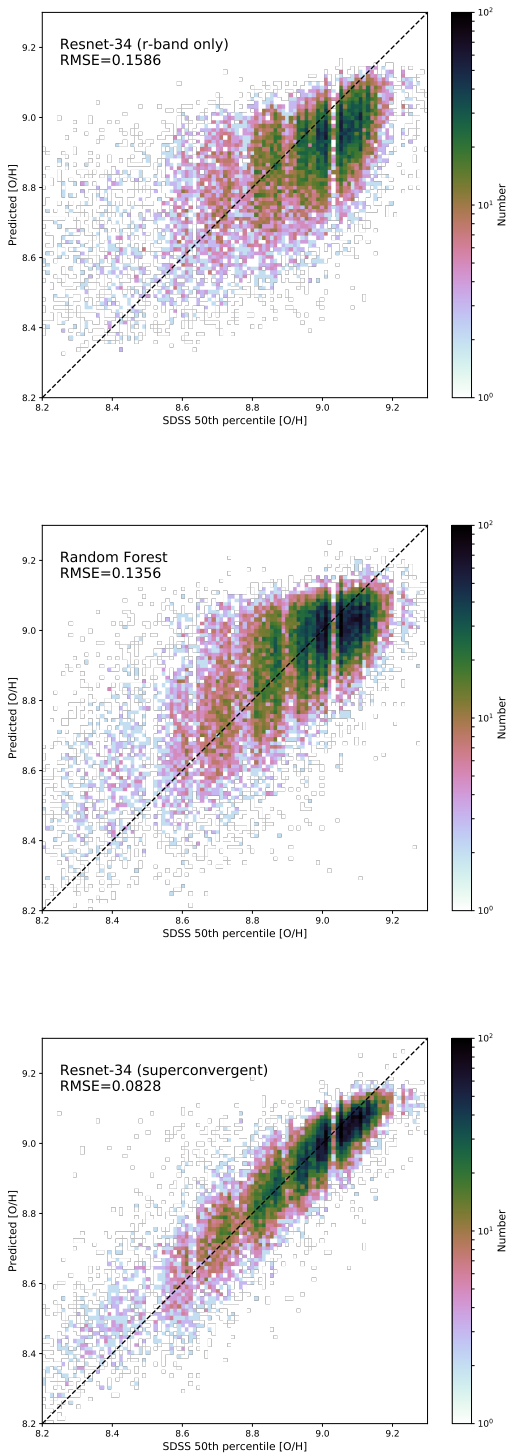

Figure 2. SDSS 50th percentile values for oxygen abundance derived from spectroscopy versus trained predictions for a variety of machine learning methods. (Upper) 34-layer Resnet predictions using only r-band imaging. (Center) Random forest predictions using only *gri* photometry. (Lower) 34-layer Resnet (superconvergent training) using *gri* imaging.

4.2 Comparisons to Previous Works

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Obviously compare with Huertas-Company et al. Also look at Viviana's work as benchmark. Compare with Galaxy Zoo as well. Maybe CMU Deeplens paper for finding Einstein rings/arcs. Note work from Sara Ellison's group: 2016MNRAS.455..370E, 2017MNRAS.464.3796T, 2016MNRAS.457.2086T

5 THE MASS-METALLICITY RELATION

Given the low scatter in metallicity residuals (predicted − true), we consider if the CNN is able to accurately predict stellar mass, and then leverage the mass-metallicity relation (MMR; Tremonti et al. 2004) to infer metallicity. Alternatively, if SFR and M_* are learned, then the fundamental metallicity relation (FMR; Mannucci et al. 2010, Lara-Lopez et al. 2010) may explain low residuals.

If there existed a fourth parameter, perhaps morphological in nature, then the marginalized MMR or FMR over particular values of this fourth parameter should be an even tighter relationship. Such a result would be analogous to how the MMR at any controlled star formation rate (SFR) has smaller scatter than over all SFRs.

We find that the MMR constructed using SDSS-measured metallicity (via R_{23}) and the MMR constructed using CNN-predicted metallicity (from *gri* imaging) are quantitatively different. In both cases we have used the GalSpecExtra catalog for stellar masses. The MMR using CNN metallicity has smaller scatter, where the weighted average ratio of scatter is NMAD(CNN)/NMAD(SDSS) = 0.821 ± 0.077.

We also produced plots of the MMR using SDSS- and CNN-predicted masses, and SDSS metallicity from the GalSpecExtra catalog. Here the scatter in metallicity at given mass is again lower when using CNN predictions than for SDSS measurements. The scatter is smaller by a factor NMAD(CNN)/NMAD(SDSS) $= 0.811 \pm 0.057$.

It is possible that the metallicity depends on some morphological component in addition to the emission lines informing the spectroscopic measurement. However, we believe that the qualitatively accurate MMR is actually artifact of the CNN's limitations rather than its strengths. Instead of finding a more fundamental representation of metallicity, the CNN likely detects the stellar mass. In the case of using our CNN to predict metallicity, the slope of the MMR is shallower than from the "true" data. However, when the CNN is used to predict mass, the slope of is too steep. This effect is driven by the fact that the CNN is unable to predict the full range of metallicities or masses, since the fraction of training examples at extremely low or high values is small. **Therefore, the scatter in all of its predictions is narrower, such that when it predicts metallicity, the**

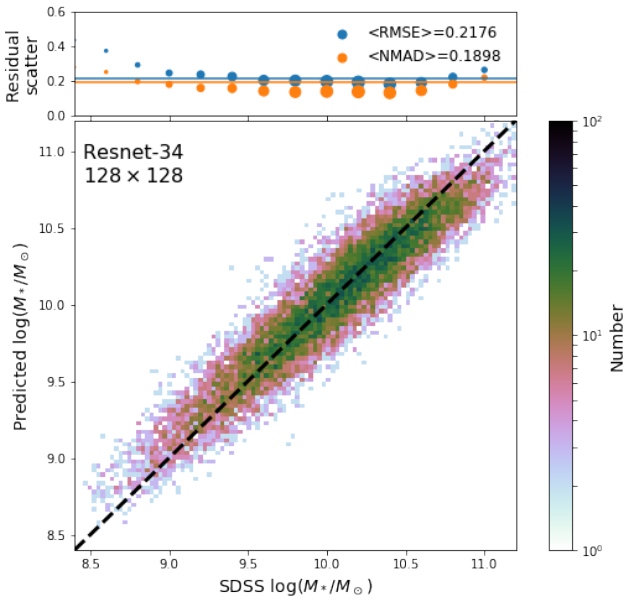

Figure 3. Same as Figure 1, except that stellar mass predictions and truths are being compared (citation?).

truncated range forces the MMR relationship to appear shallower, and when the CNN predicts M_*, the limited range in mass cause the MMR relationship to appear steeper.

Why then does the MMR hold at all? For if the CNN predicts, e.g., too narrow a distribution of metallicity, then what prevents those predictions from scattering over the full range in stellar mass? The answer is quite possibly that the CNN only "sees" the metallicity via the stellar mass, such that the predicted MMR is effectively a relationship between the predicted mass and the true mass. This theory is bolstered marginally by the fact that the fraction scatter for predicting mass is lower than for predicting metallicity (albeit not significantly so). More intuitively, we might believe that the CNN can "see" that read and dead elliptical galaxies are higher in stellar mass, and that blue irregulars are lower in stellar mass. Then the CNN simply needs to propagate the mass prediction through the tight ~ 0.1 dex MMR.

How then can we achieve a NMAD = 0.067 dex in metallicity by noisily propagating a signal with NMAD = 0.22 scatter through the MMR which has 0.1 dex intrinsic scatter (Tremonti et al. 2004)?

M/L ratio of redder galaxies are higher at fixed luminosity (Bell & de Jong 2001; Kauffman+03).

6 FUTURE APPLICATIONS

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

7 SUMMARY

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Our main conclusions are the following:

(i) Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

(ii) Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

ACKNOWLEDGEMENTS

The authors also wish to thank the anonymous referee whose comments and suggestions significantly improved both the quality and clarity of this work. The authors also thank David Shih and Matthew Buckley for use of their GPU cluster at Rutgers University High Energy Experimental Physics department. This research made use of the IPython package (**?**) and matplotlib, a Python library for publication quality graphics (**?**). Funding for the SDSS and SDSS-II has been provided by the Alfred P. Sloan Foundation, the Participating Institutions, the National Science Foundation, the U.S. Department of Energy, the National Aeronautics and Space Administration, the Japanese Monbukagakusho, the Max Planck Society, and the Higher Education Funding Council for England. The SDSS Web Site is http://www.sdss.org/.

APPENDIX A: RESIDUAL CONVOLUTIONAL NEURAL NETWORKS

In general, CNNs become difficult to train after many layers are added. We select an architecture called a residual neural network, or *resnets* (He et al. 2015), which contains enhanced "shortcut connections" but are otherwise similar to other CNNs. Resnets have been shown to continue learning with increasing depth without the added cost of extra parameters. This is likely because the network has already found the best representation possible at a shallower layer, and further layers simply noisily propagate the same signal and degrade the loss. Therefore resnets are able to learn deep representations of the data.

We find that a 34-layer resnet (Resnet-34) architecture can be trained efficiently on a Pascal P100 with 16 GB of memory. Using the hyperparameters described below, an epoch takes about 60 seconds to train. We initialize our resnet with pretrained weights from the ImageNet (Russakovsky et al. 2014; He et al. 2015) 1.7 million image data set trained to recognize 1000 classes of objects found on Earth (i.e., cats, dogs, or cars). In practice, the filters learned through the early layers of the pretrained CNN can be used for other image recognition tasks, aptly named "transfer learning" (cite).

A1 Hyperparameter selection

A2 Data augmentation

Nearly all neural networks benefit from larger training samples because they help prevent overfitting. Outside of the local Universe, galaxies are seen at nearly random orientation; such invariance permits synthetic data to be generated from rotations and flips of the training images. Each image is fed into the network along with four augmented versions, thus increasing the total training sample by a factor of five. This technique is called data augmentation, and is particularly helpful for the network to learn uncommon or unrepresented truth values (e.g., in our case, very metal-poor or metal-rich galaxies). Each training-augmentation is fed-forward through the network and gradient contributions are computed together as part of the same mini-batch. A similar process is applied to the network during predictions, which is called test-time augmentation (TTA). Synthetic images are generated according to the same rules applied to the training data set. The CNN predicts an ensemble average over the augmented images. It has been found that data augmentation improves predictions by as much as $\sim$ 20% (cite).

A3 Cyclical learning rate annealing

The learning rate determines how large of a step each network weight takes in the direction of the backpropagated error. A large learning rate thus forces the weights to make large updates, which generally prevents overfitting but also may cause the parameters to overshoot minima in the loss function landscape. A small learning rate may allow the weights to get stuck in a local minima near their initial positions, or also might cause the network to learn very slowly. The number of local minima increases exponentially with dimensionality (cite), so selecting a low learning rate does not ensure that the network will eventually make it to near the global minimum. Therefore, it is often useful to anneal, or reduce, the learning rate from a high value in the beginning – which allows the weights to find the right ballpark values – to a low value – which allows the weights to take finer steps near the global minimum – over the course of multiple training epochs.

In practice, annealing can be enforced manually, e.g., the learning rate might be reduced by a factor of 10 every time the loss function plateaus over a number of epochs. Eventually even reduced learning rates do not permit additional improvement of the loss, and so the training period is concluded. We instead use a method called cosine annealing, during which the learning rate is annealed continuously over one or more epochs for *each* mini-batch until it eventually reaches zero.

We employ stochastic gradient descent with restarts (SGDR), over progressively longer training cycles, and restart cosine annealing over each cycle (Leslie Smith 2015?). For example, the first cycle comprises one epoch during which the learning rate is cosine annealed. It then trains for another cycle with cosine annealing, which starts at the same learning rate but is annealed over twice the duration (two epochs). These "restarts" have been shown to kick the weights configuration out of saddle points in the loss of some high-dimensional parameter space. Training can then proceed past what appear to be local minima but are actually saddle points (where gradient descent generally performs poorly).

A4 Layer learning rates

ADD MORE HERE

Frozen training to get final activates in right "ballpark." We then unfreeze all layers and train using different rates in different layer groups.

A5 Batch normalization and Dropout

Batch normalization (BN; 1502.03167) is a technique developed to fix the vanishing gradients problem which make deep networks inefficiently slow to train. The issue arises when gradients are backpropagated through deep neural networks to update weights, and loss contributions become vanishingly small except only when the weights have small magnitudes. Normalizing the inputs to each mini-batch mean and standard deviation somewhat remedies the problem, and has been applied to (convolutional) neural networks since the 1990s. BN extends this normalization to all activations in hidden layers, thus adding two hyperparameters which modulate the mean and standard deviation of each layer to zero and unity, respectively. The BN hyperparameters are learned for each mini-batch and are updated in addition to weight parameters during the backward pass.

Without BN, activations in any given layer may span a large range and contributions from certain parts of the network may become dwarfed by others. After the normalization step, all pre-activations are on the same scale and thus gradient descent steps can more efficiently traverse the loss function space. Training proceeds more rapidly and converges toward a solution more quickly. Choice of mini-batch size also impacts the learning rate, as small batch size increases stochasticity in each gradient. Large batch size allows for better parallelization on the GPU and ensures smoother gradients. We note that smooth gradients across mini-batches can prevent the solution from hopping out of local basins, and negatively impact performance. Increasing the learning rate as batch size is increased can resolve this problem, at least in part, but limits the convergence ability of the network (until the rate is annealed). We find in practice that batch sizes between 128 and 512 work best at balancing noisy gradients and learning rate.

Dropout is a method of disabling a random subset of connections after linear layers for each mini-batch (Hinton

et al. 2012). By removing random connections between fully-connected layers, dropout effectively treats each mini-batch as one in an ensemble of random training subsets . It is instructive to think of each mini-batch in the full training data as analogous to a decision tree in a random forest. The ensemble of learned gradients is less prone to overfit the training data set because the network is forced to discard random (and potentially valuable) information. The resulting network is better able to, for example, learn subtle differences in the data that would otherwise be ignored when more obvious features dominate the gradient descent process. Since our Resnet architecture is broadly separated into multiple groups of layers, we can apply lower dropout rate at earlier layers in order to ensure that those filters are learned more quickly. Using a differential dropout rate is sensible because filters learned in the first few layers of the network tend to be Gabor filters and edges in general, and there is little risk of overfitting when such low-level abstractions are always necessary for learning high-level features in the middle and final layers.

During validation and testing, dropout is not used because all of the data is useful for predictive power. We use dropout rates of 0.25 for the linear layer after the early group, and 0.50 at the later linear layer. We find that dropout combined with BN – although not recommended in the original paper – work in tandem to boost training speeds and avoid overfitting.

This paper has been typeset from a TEX/LATEX file prepared by the author.