MNRAS **000**, 1–5 (2018) Preprint 28 June 2018 Compiled using MNRAS LATEX style file v3.0

Predicting Galaxy Metallicity with Three-Color Images and 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

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.

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.

2.1 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).

2.2 Hyperparameter selection

2.3 Training the network

2.3.1 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).

2.3.2 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).

2.3.3 Differential learning

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.

2.3.4 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.

2.3.5 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). Superconvergent 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()
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.

3 DATA

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.

```
SELECT top 100
  G.objID,
  GSI.specObjID,
  G.ra,
  G.dec,
  S.z,
  S.zErr,
  S.velDisp,
  S.velDispErr,
  G.modelMag_u,
  modelMagErr_u,
  G.modelMag_g,
  modelMagErr_g,
  G.modelMag_r,
  modelMagErr_r,
  G.modelMag_i,
  modelMagErr_i,
  G.modelMag_z,
  modelMagErr_z,
  G.petroR50_r,
  G.petroR90_r,
  GSL.nii_6584_flux,
  GSL.nii_6584_flux_err,
  GSL.h_alpha_flux,
  GSL.h_alpha_flux_err,
  GSL.oiii_5007_flux,
  GSL.oiii_5007_flux_err,
  GSL.h_beta_flux,
  GSL.h_beta_flux_err,
  GSL.h_delta_flux,
  GSL.h_delta_flux_err,
  GSX.d4000,
  GSX.d4000_err,
  GSE.bptclass,
  GSE.oh_p2p5,
  GSE.oh_p16,
  GSE.oh_p50,
  GSE.oh_p84,
  GSE.oh_p97p5,
  GSE.lgm_tot_p50,
  GSE.sfr_tot_p50
INTO mydb.SDSSspecgalsDR14
FROM SpecObj as S
  JOIN Galaxy as G
  ON G.ObjID = S.bestObjID
  JOIN GalSpecLine as GSL
  ON GSL.specObjID = S.specObjID
  JOIN GalSpecInfo AS GSI
  ON GSI.specObjID = S.specObjID
  JOIN GalSpecIndx AS GSX
  ON GSX.specObjID = S.specObjID
  JOIN GalSpecExtra AS GSE
  ON GSE.specObjID = S.specObjID
WHERE (G.petroMag_r > 10 AND G.petroMag_r < 18)
  AND (G.modelMag_u-G.modelMag_r) > 0
  AND (G.modelMag_u-G.modelMag_r) < 6
  AND (modelMag_u > 10 AND modelMag_u < 25)
  AND (modelMag_g > 10 AND modelMag_g < 25)
AND (modelMag_r > 10 AND modelMag_r < 25)
AND (modelMag_i > 10 AND modelMag_i < 25)
AND (modelMag_z > 10 AND modelMag_z < 25)
AND S.rChi2 < 2
AND (S.zErr > 0 AND S.zErr < 0.01)
AND S.z > 0.02
AND GSE.oh_p50 != −9999
```

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.

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 DISCUSSION

5.1 Mass metallicity relation + fundamental plane

Are we seeing this correlation? Discuss covariance of errors in predictions

Are we able to explain some of the scatter in the fundamental plane? Is our scatter *lower* than that of the fundamental plane?

5.2 How important is morphology relative to color?

If the variance were to *independently* scale with contributions from morphology and multiple photometric bands, then we would expect the MSE (*or wait, actually the variance?*) to decrease like:

$$\mathrm{MSE}_{\mathrm{gri}}^{-1} \approx \mathrm{MSE}_{\mathrm{photo}}^{-1} + \mathrm{MSE}_{\mathrm{u}}^{2} + \mathrm{MSE}_{\mathrm{r}}^{2} + +\mathrm{MSE}_{\mathrm{r}}^{2} \qquad (1)$$

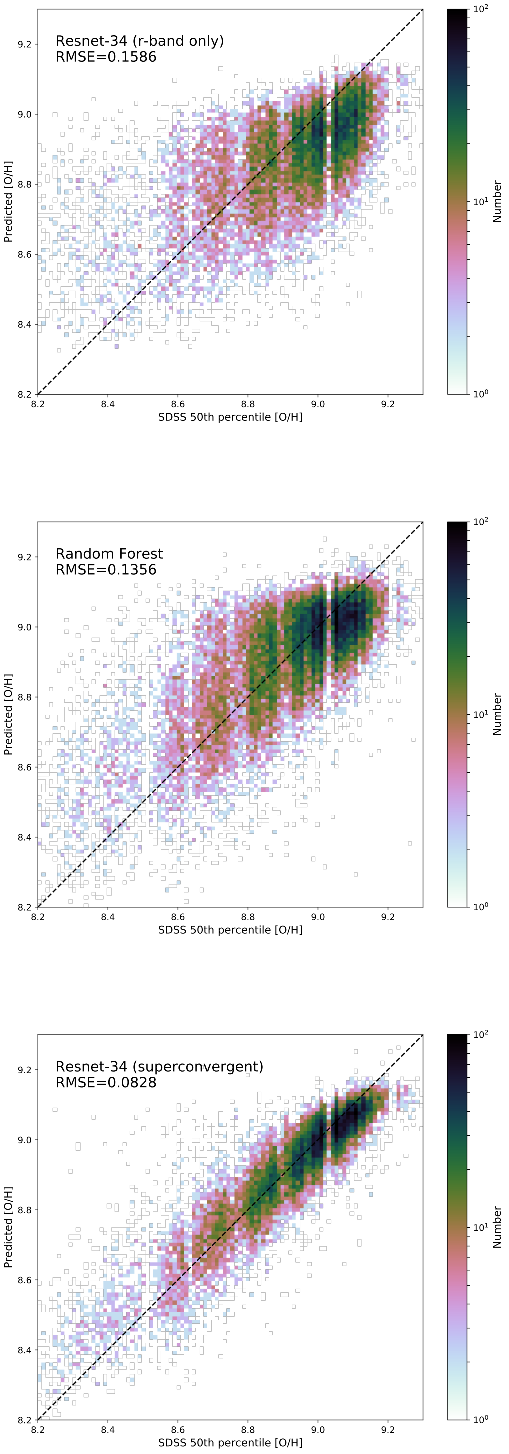

Figure 1. SDSS 50th percentile values for oxygen abundance derived from spectroscopy versus trained predictions for a variety of machine learning methods. (Upper) 34-layer Resnet predictions using only r-band imaging. (Center) Random forest predictions using only gri photometry. (Lower) 34-layer Resnet (superconvergent training) using gri imaging.

Actually, more properly:

$$\sigma_{\rm gri}^{-2} = \sigma_{\rm photo}^{-2} + \sigma_{\rm g}^{-2} + \sigma_{\rm r}^{-2} + \sigma_{\rm i}^{-2} - (\rho_{\rm gr}\sigma_{\rm g}\sigma_{\rm r})^{-1} \\ - (\rho_{\rm gi}\sigma_{\rm g}\sigma_{\rm i})^{-1} - (\rho_{\rm ri}\sigma_{\rm r}\sigma_{\rm i})^{-1}, \qquad (2)$$

where we have assumed that g, r, and i images are correlated with correlation coefficients given by ρ. We may actually be able to estimate ρ by taking pairwise cross-correlations of different band images, e.g.,

$$\rho_{\mu\nu} = \frac{I_\mu \star I_\nu}{\sqrt{(I_\mu \star I_\mu)\,(I_\nu \star I_\nu)}}. \qquad (3)$$

Here I have represented a correlation using the $\star$ symbol.

Also do: Compare r-band vs. gr (two-color) vs. gri three-color imaging.

5.3 Does morphology explain some of the scatter in the fundamental plane?

5.4 Can stellar population age explain some of the scatter in the fundamental plane?

5.5 Why do CNNs work so well at predicting spectroscopic quantities?

Can we find a non-CNN parameter or method of explaining the scatter?

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 (include this only after re-submission). The authors also thank David Shih and Matthew Buckley for use of their GPU cluster at Rutgers University High Energy Experimental Physics department, and Kartheik Iyer for providing SED fits as benchmark as well as for valuable discussion. 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/.

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