GANs for Data Augmentation

Eric Gentry and Jiahua You

Link to report;
link to github repo

Goal:

1. Split the data into **training and testing sets**.

2. Build one **classifier that is trained on the real data**, using traditional data augmentation transformations (reflections, translations, etc). This will be the **benchmark**.

3. **Build a conditional GAN**, which accepts the metadata features “distance of galaxy” and “mass of galaxy”, and generates “realistic” images for that distance and mass. Train the GAN and then freeze its weights.

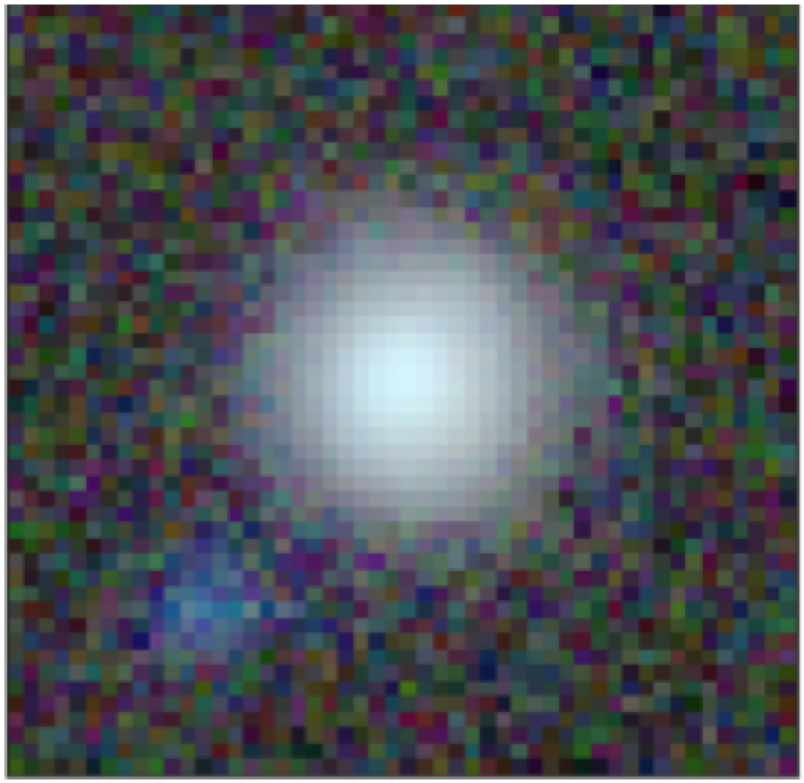
4. **Use this GAN to create training images for another classifier**, *using the same architecture* as in Step 2. This classifier is trained on purely generated images, and validated on purely real images.

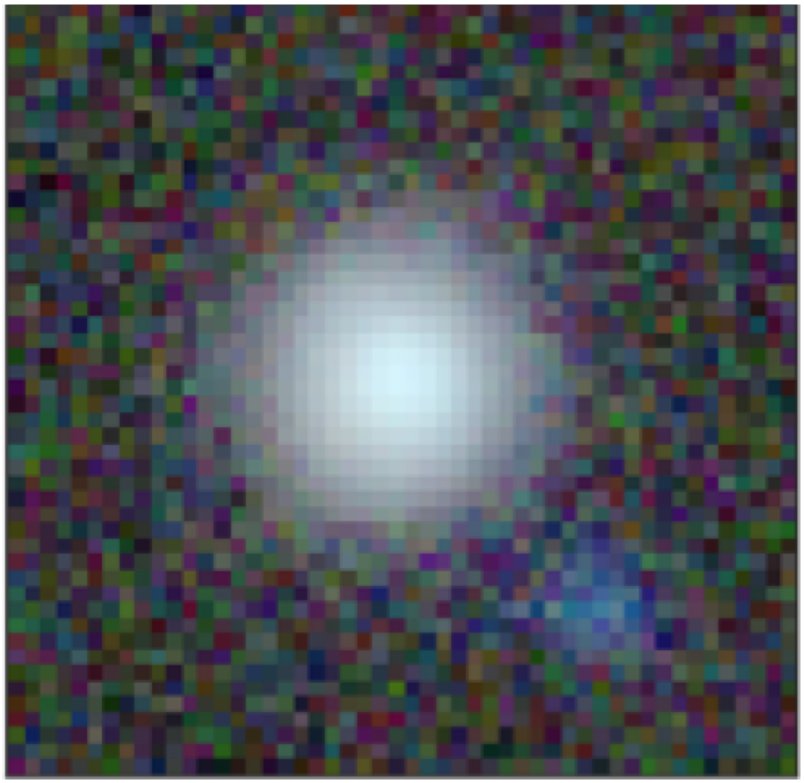
5. **Compare results** from classifier trained using traditional data augmentation (Part 2) and the classifier trained using GAN augmentation (Part 4).

Reminder: What is data augmentation

Transformations that don't change the true label

Example: Horizontal reflection

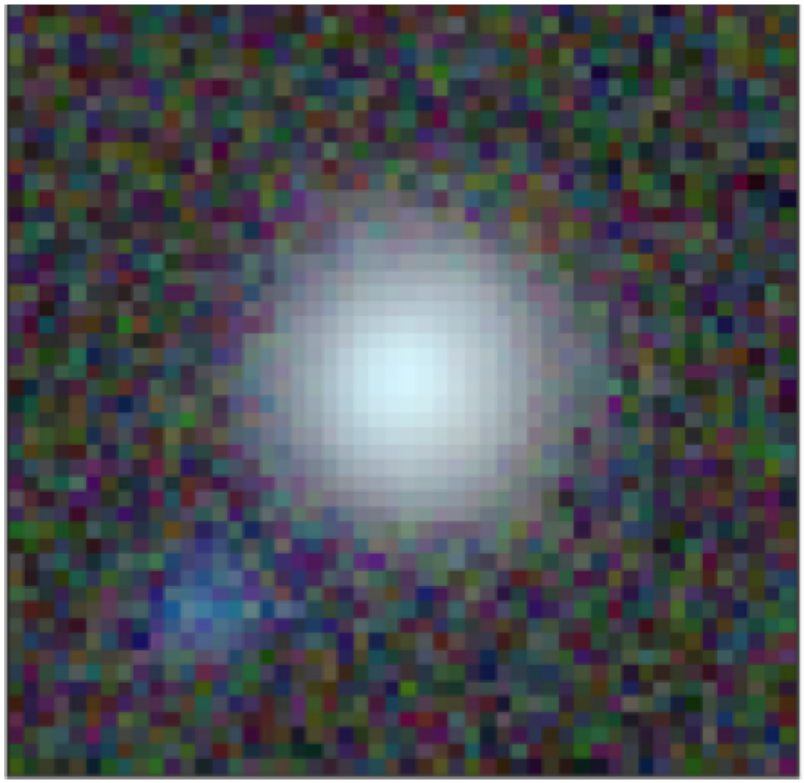


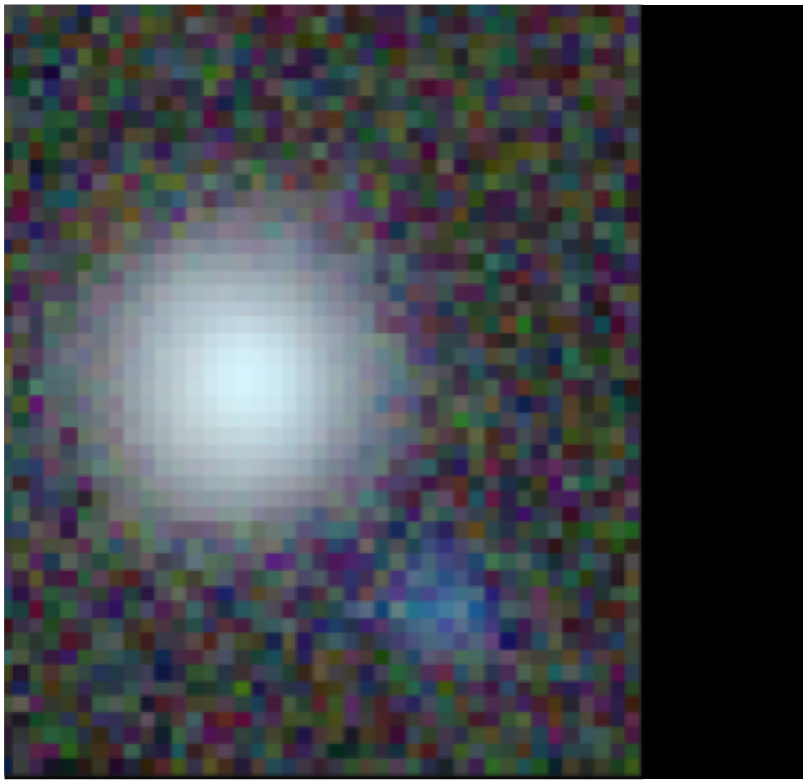


Reminder: What is data augmentation

Transformations that don't change the true label

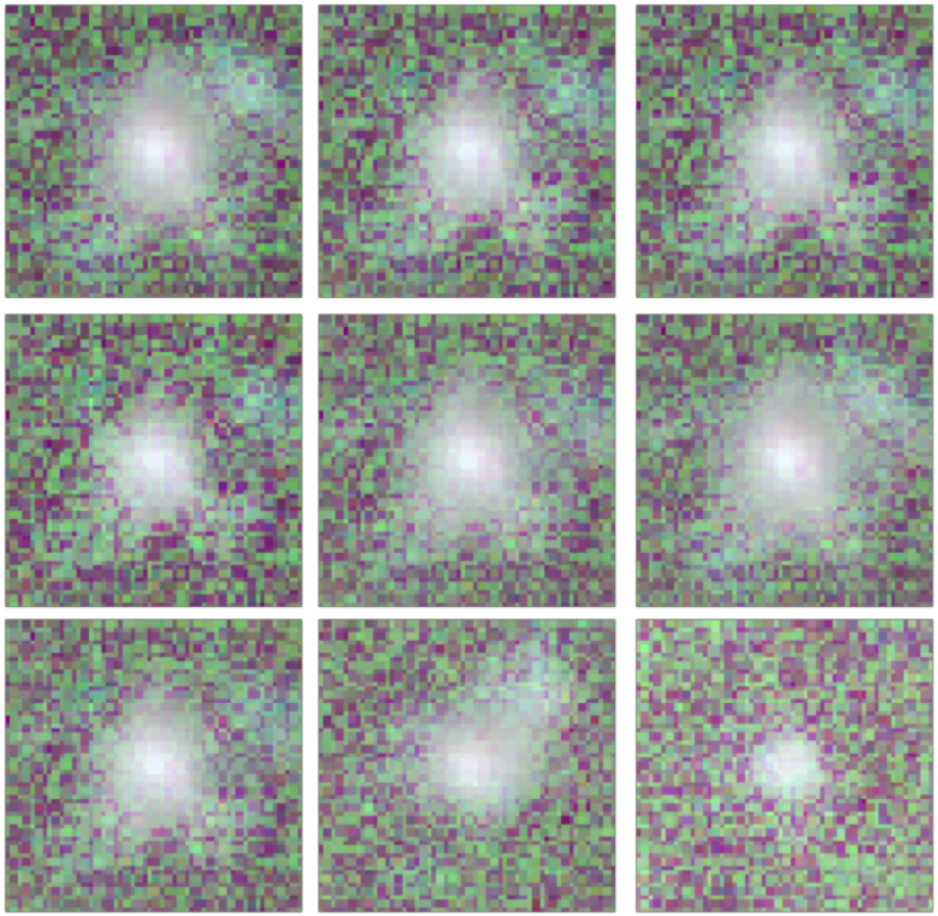
Example: Horizontal *translation* (exaggerated)





Would be much nicer if a GAN could just generate images for us

But a **reminder**: We want useful pictures, not just pretty pictures



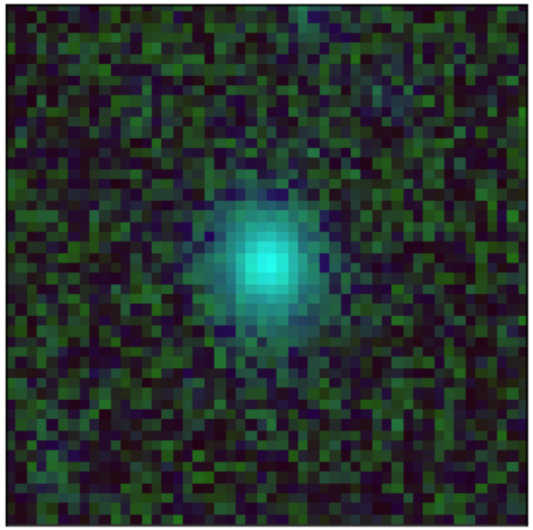
Task: classify galaxy images

Data: 2000 images * 3 channels * $(50\ px)^2$

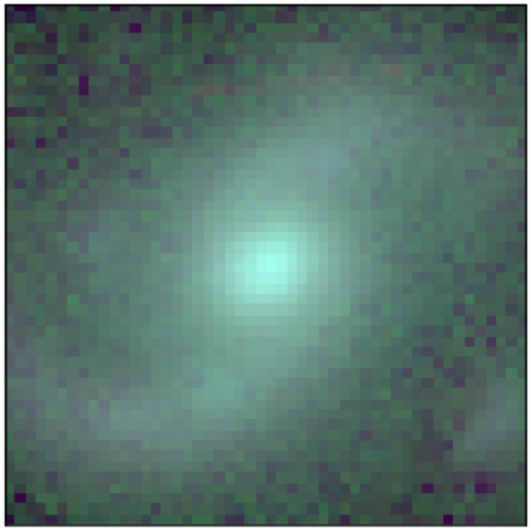
Labels:

- Distance, Mass
- “nearby low-mass galaxy” (boolean; class balance= 2:1 (False:True))

Label = True

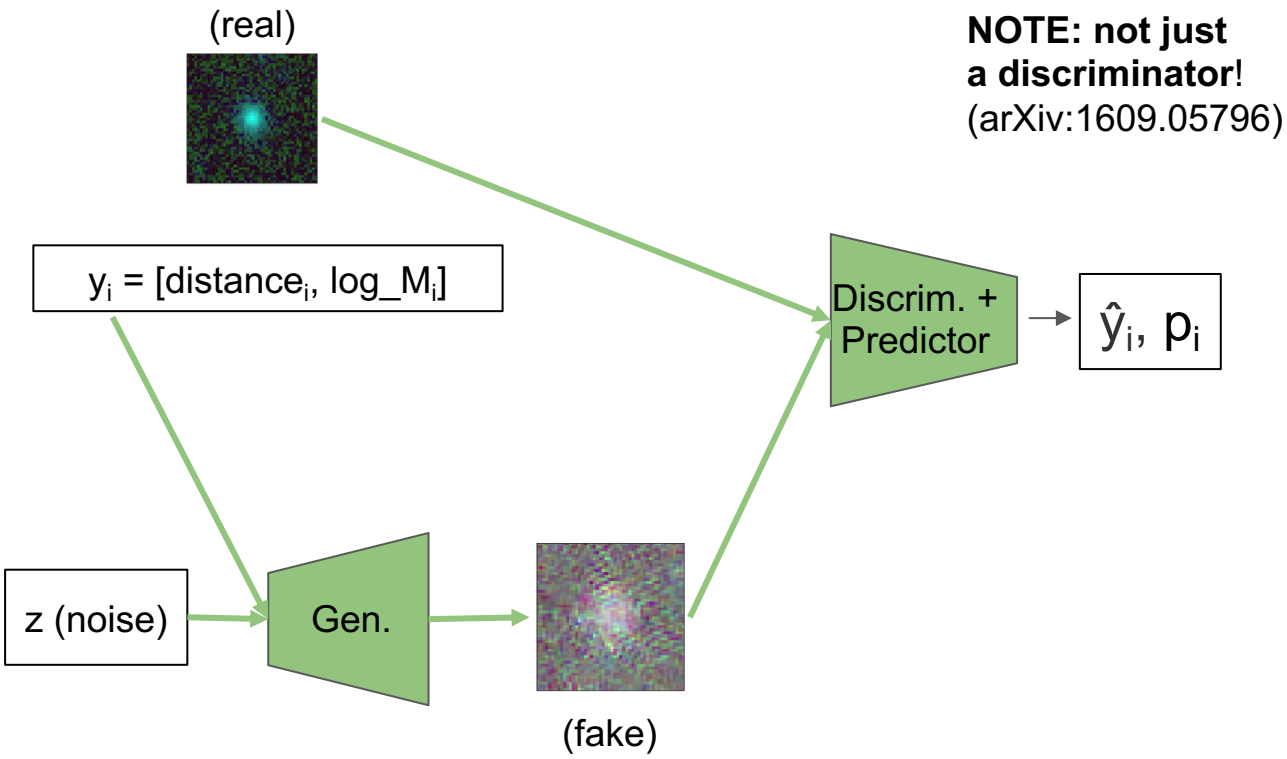


Label = False

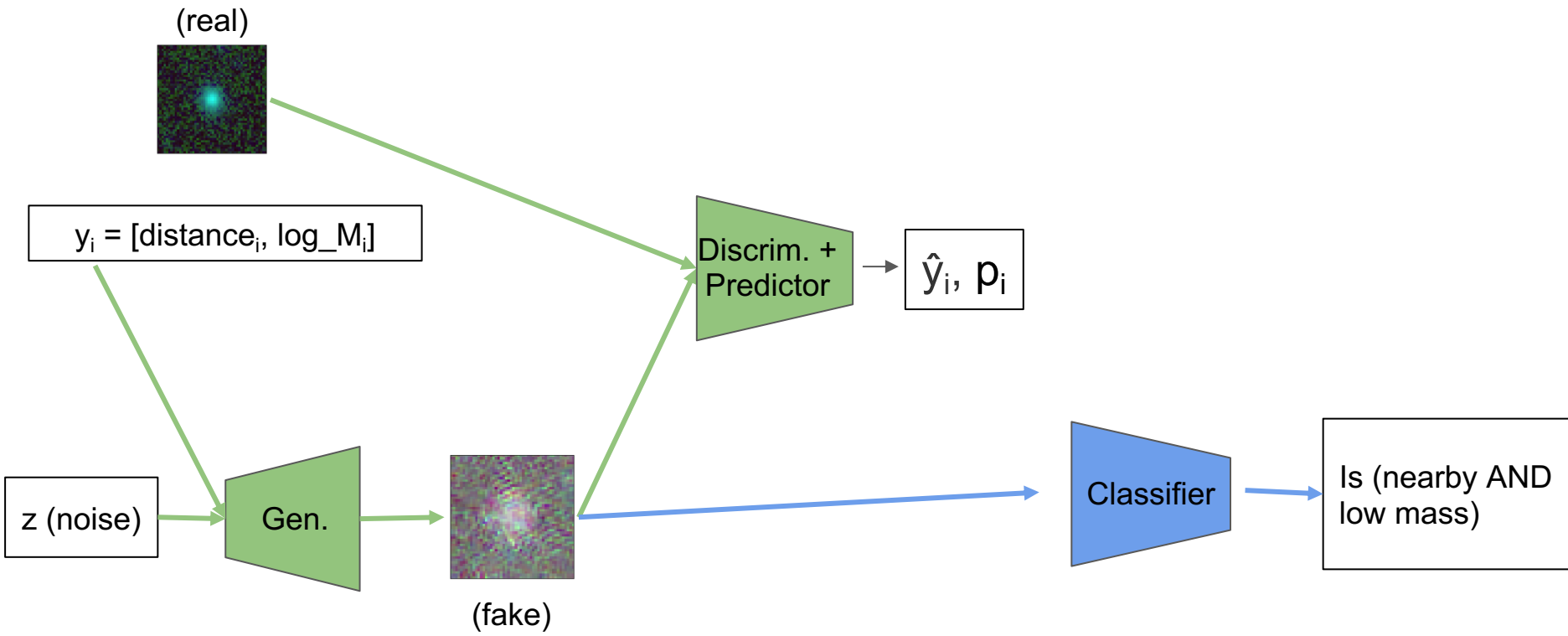


Same distance,
But 30x more massive

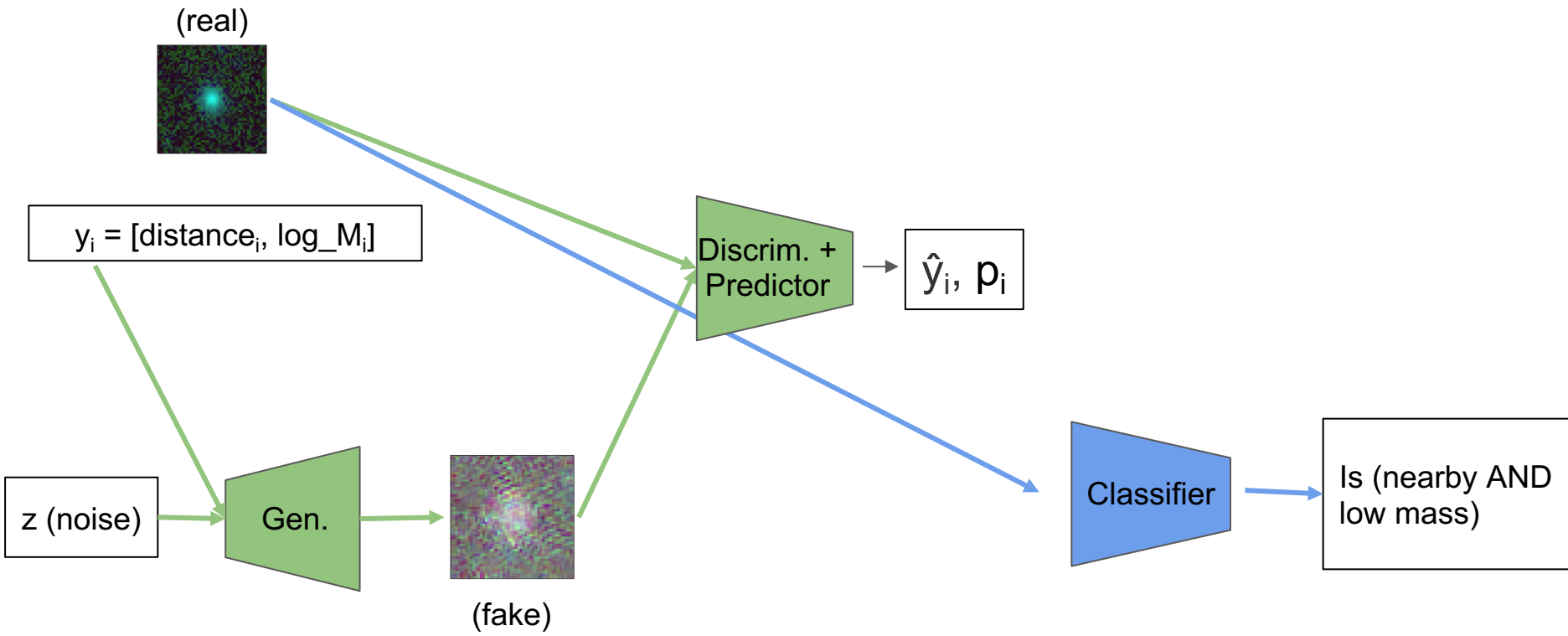
Architecture: Overview (training GAN)



Architecture: Overview (GAN ***training*** Classifier)



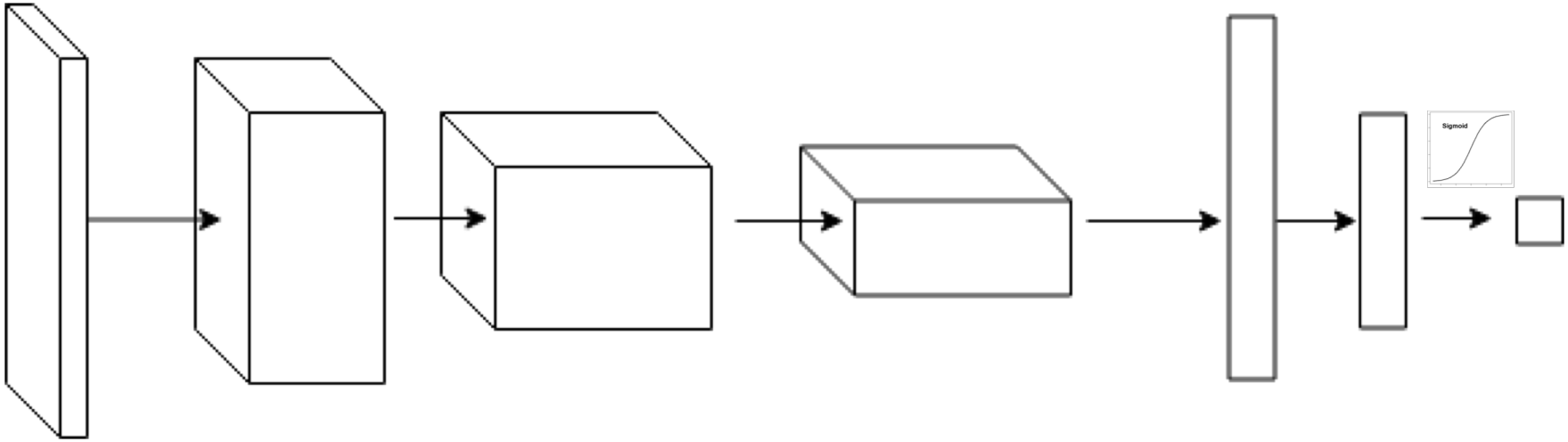
Architecture: Overview (Classifier ***validation***)



Architecture: Classifier

Conv Layers

FC Layers

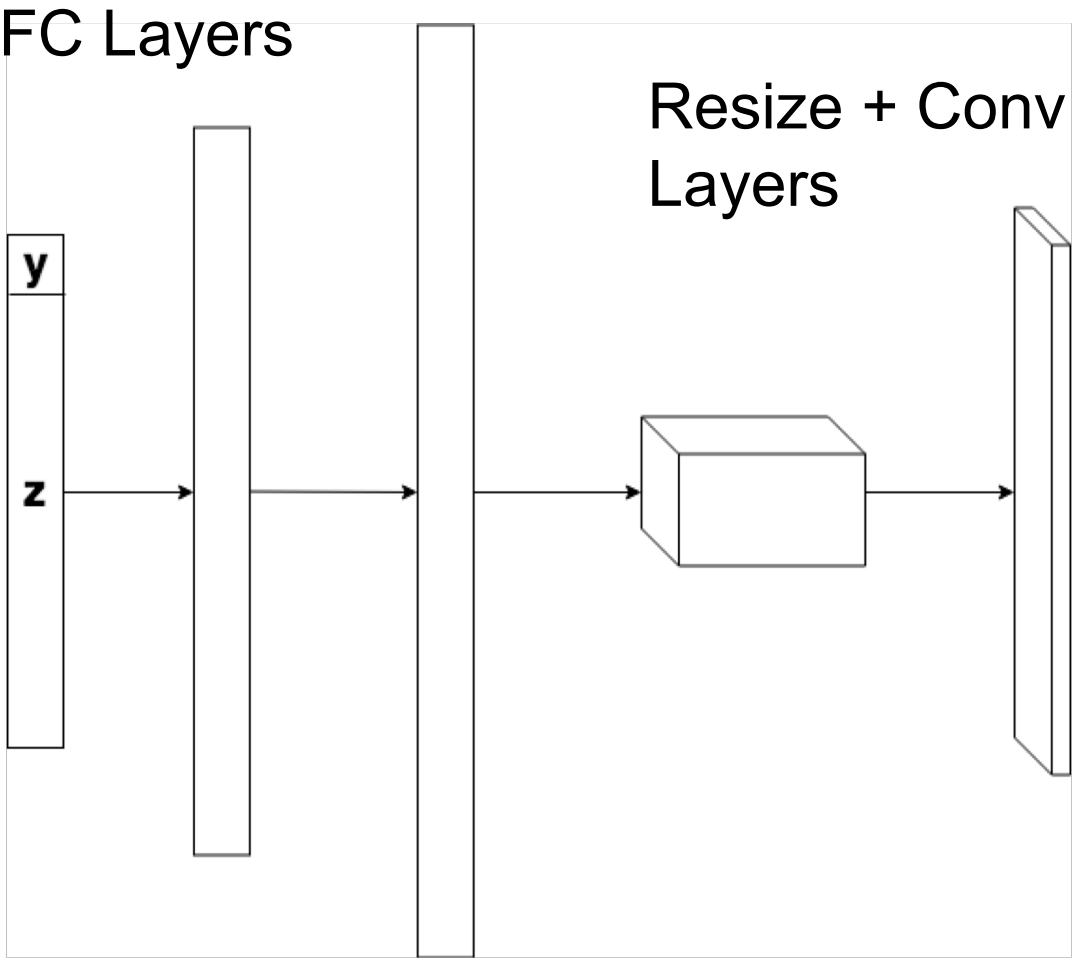


RELU activation,
MaxPool,
Dropout between layers

RELU activation,
Except sigmoid for last layer

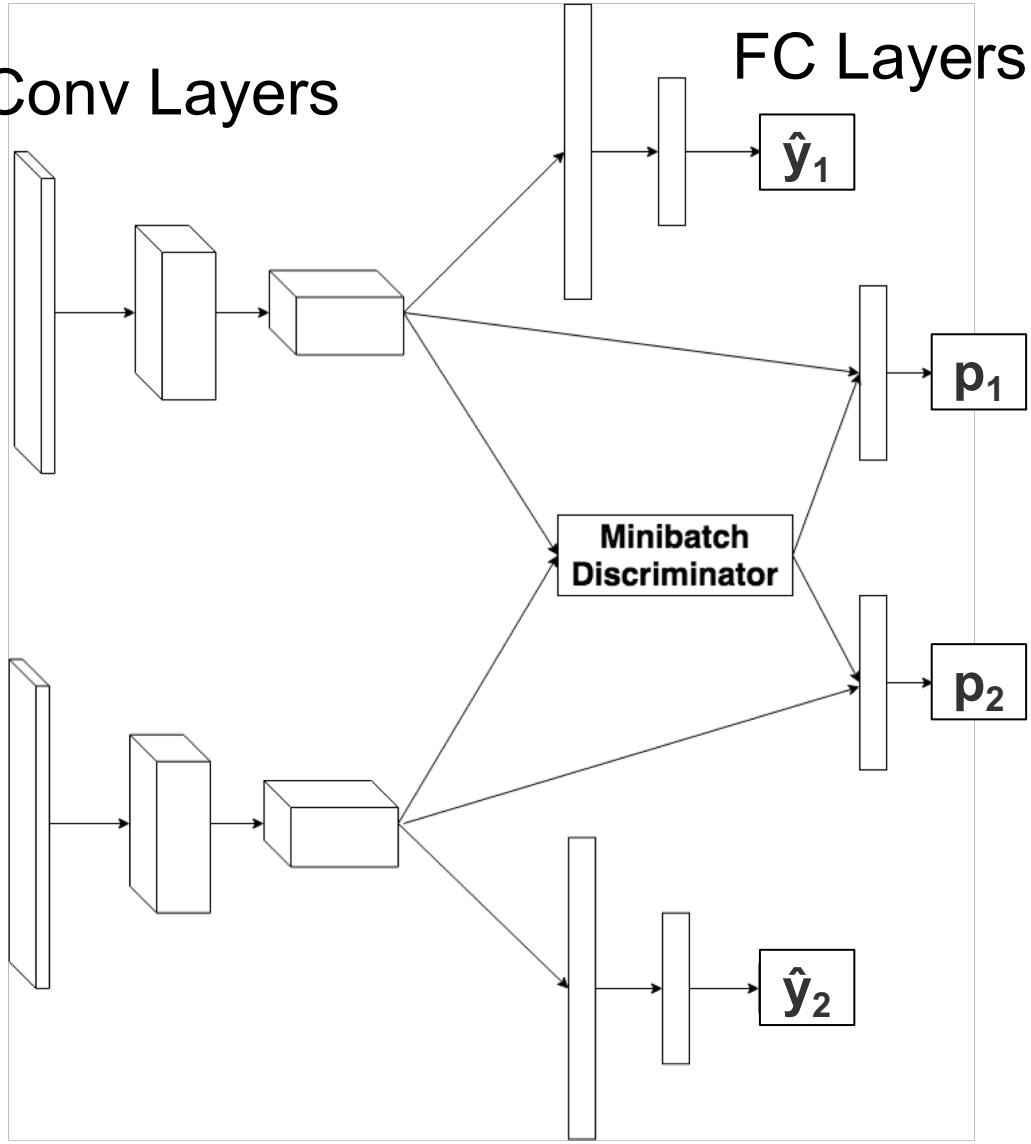
Loss function: binary cross-entropy

Architecture: GAN --- Generator



Note: not deconv. layers!
(due to checkerboard artifacts)

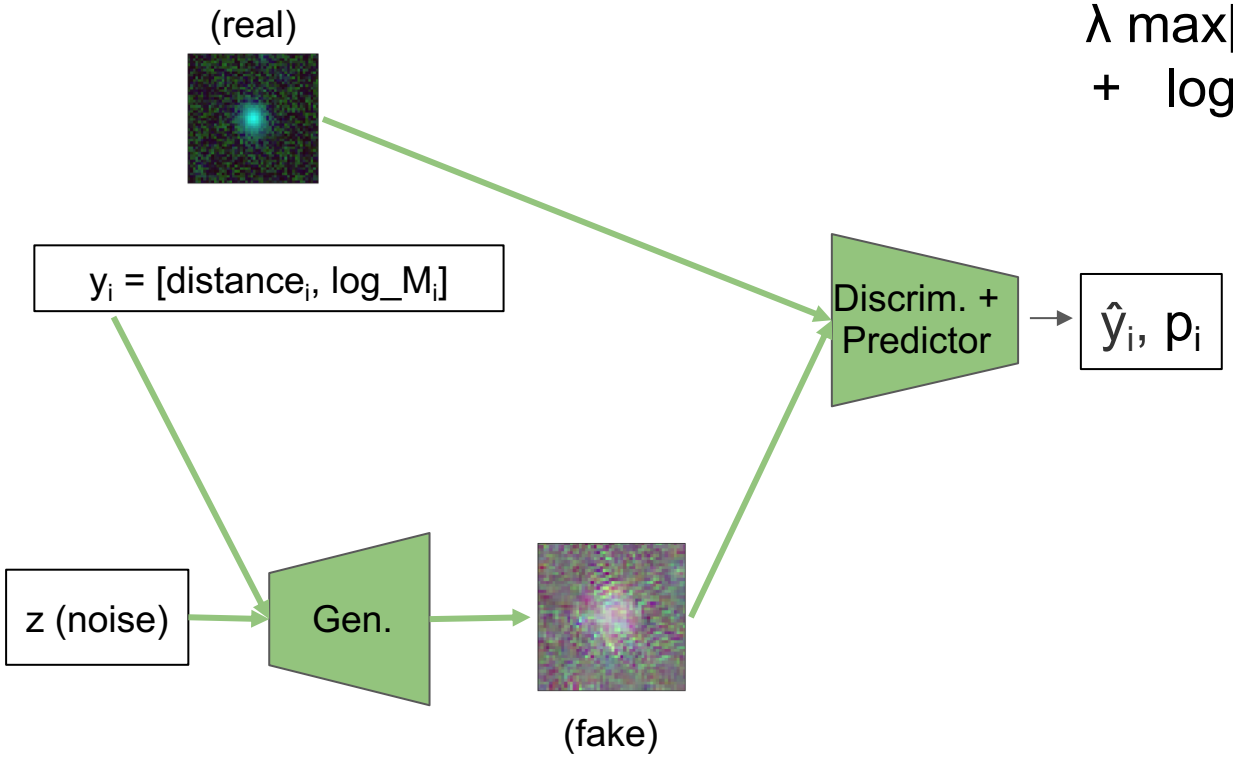
Architecture: GAN --- Discriminator/Predictor



Architecture: GAN --- Overall

Discrim./Predictor Loss:

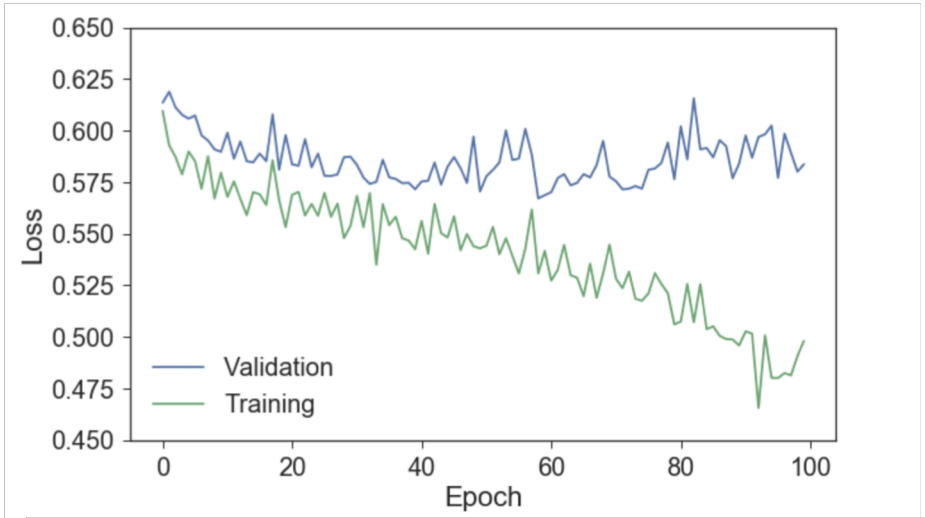
$\lambda \max[0, \ell^2(\hat{y}\_real) - \ell^2(\hat{y}\_fake)]$
$+ \quad \log(p\_real) + \log(1- p\_fake)$

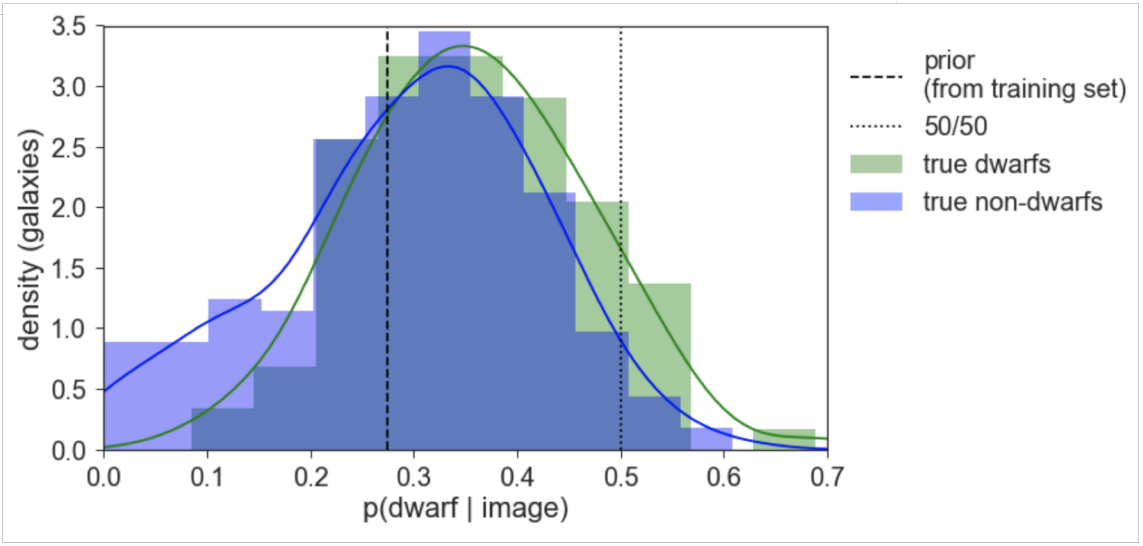


Generator Loss:

$\lambda \ell^2(\hat{y}\_fake)$
$+ \quad \log(p\_fake)$

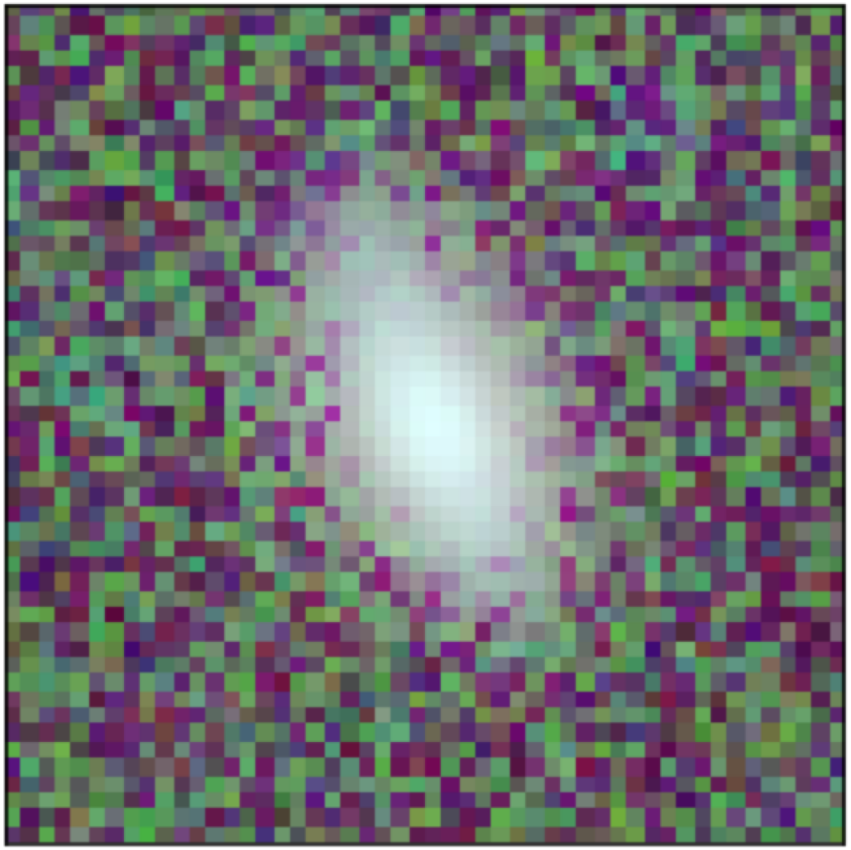
Results - Classifier Baseline



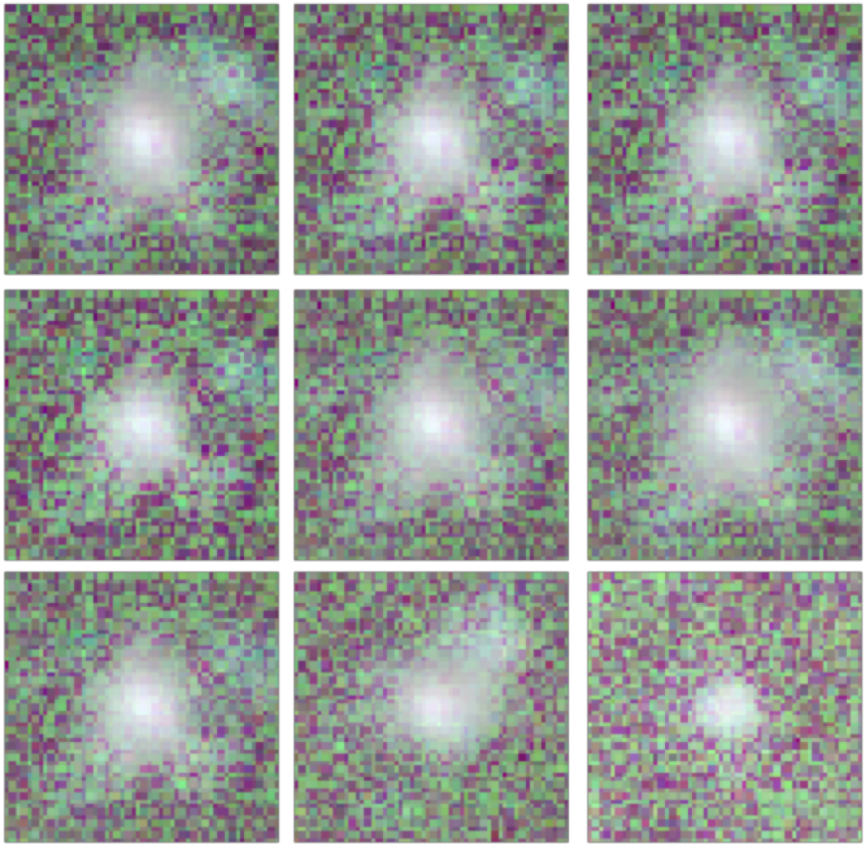


Results: GAN generated images are decent

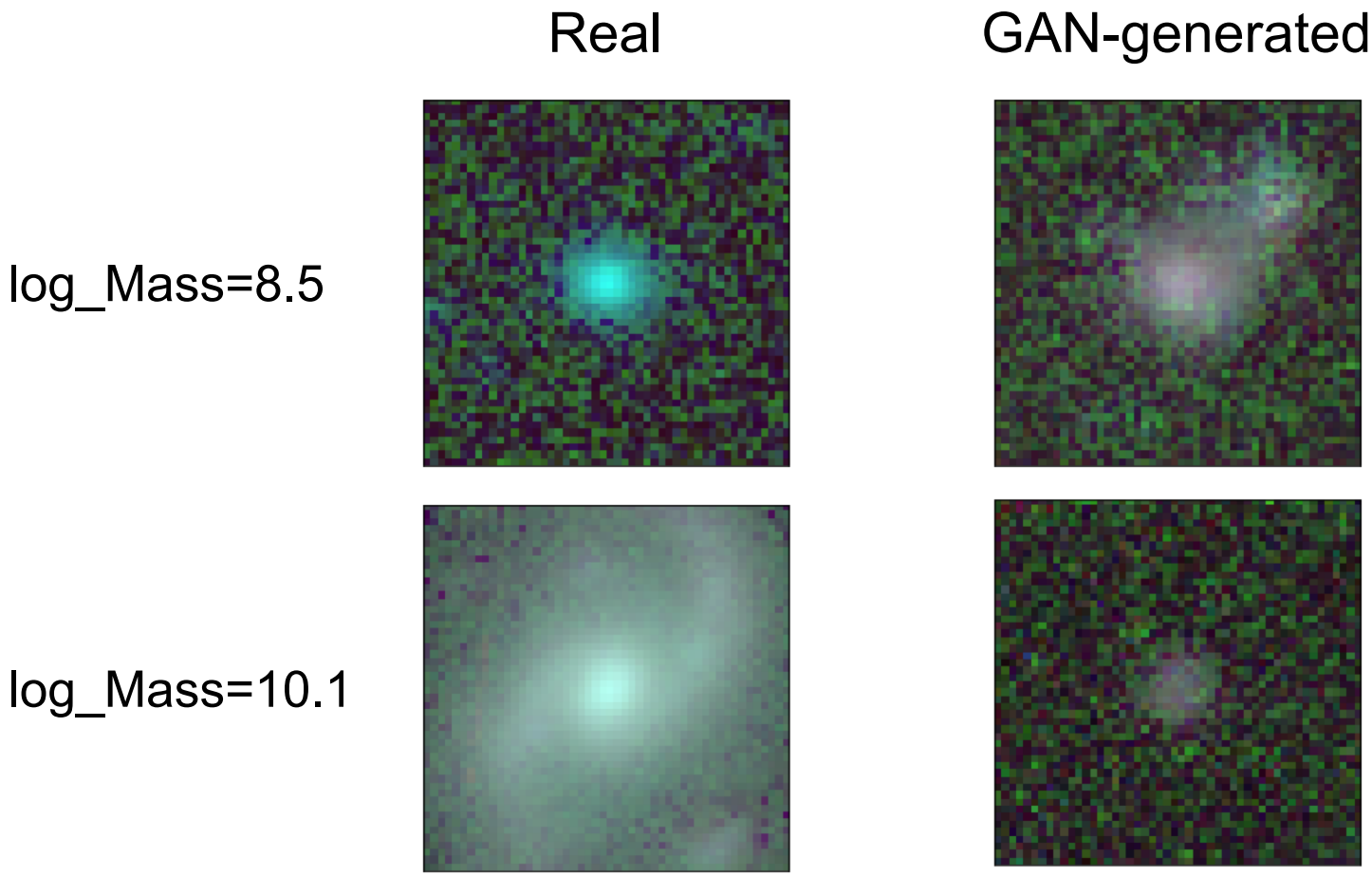
Real



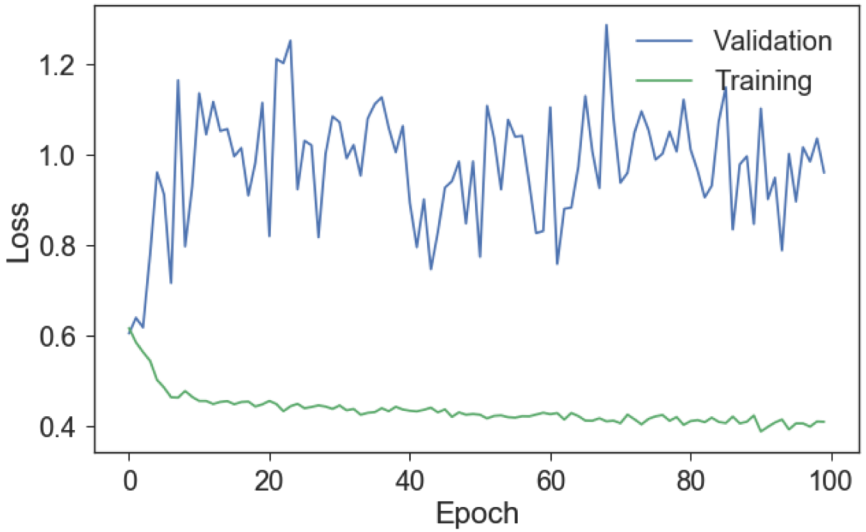
GAN-generated

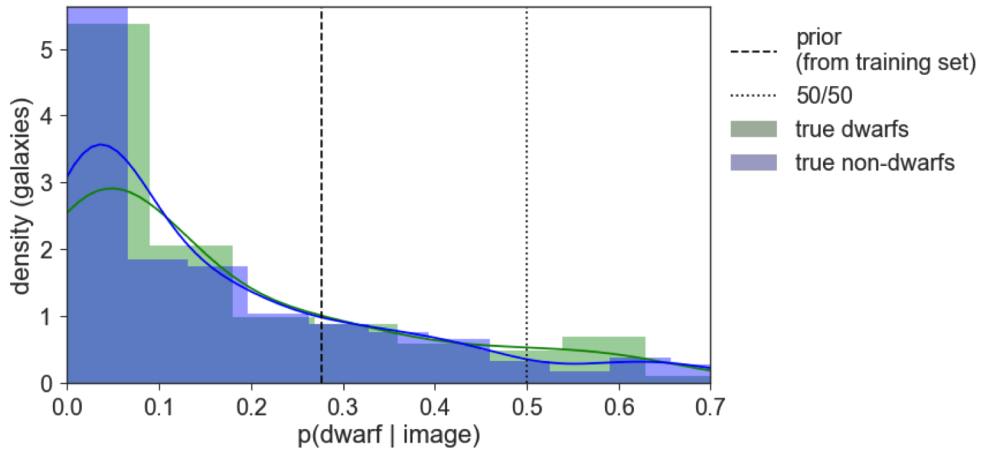


Results: GAN doesn't always capture conditional label



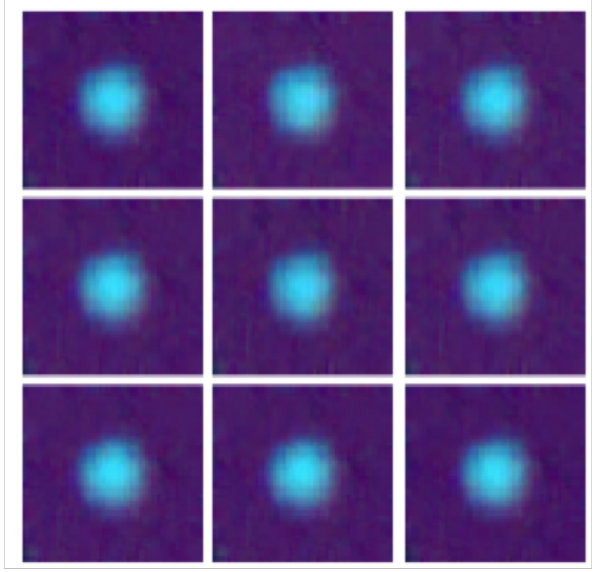
Results - Training Classifier on GAN images ***hurts*** performance





Problems encountered

1. Difficult to generate good images given **continuous conditional values**
   a. Added ***predictor*** in addition to *discriminator* (see Ravanbakhsh et al. 2016)
2. Generator **mode collapse**
   a. Partial solution: use ***minibatch discrimination***, which adds a loss for correlated generated samples (see Salimans et al. 2016)
   b. Requires balancing an l2 loss and a cross-entropy loss



9 “independent” samples from an *overtrained* GAN (without minibatch discrim.)

Conclusions

Creating Conditional-GAN is non-trivial, even for “simple” images

- Needed to hand-tune: architecture, loss functions, optimizer parameters

Training on C-GAN images decreased performance

- Probably because it doesn’t capture label-dependence properly
- If label dependence were fixed, maybe we’d get better performance by first training on GAN images, and then training on real images