CNN Regression: train on one particle, apply to another

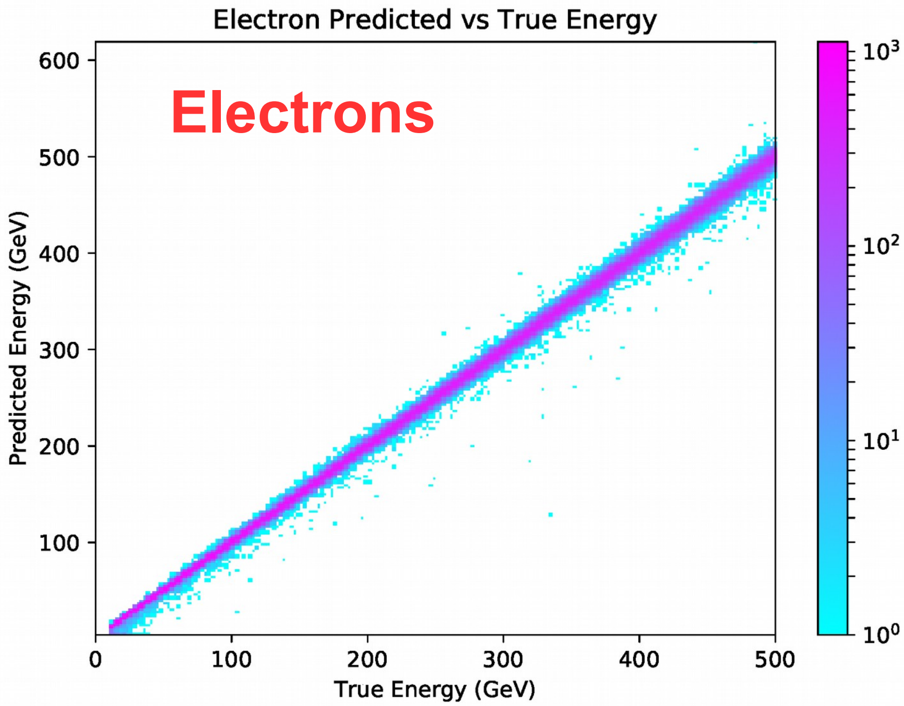
Dominick Olivito (UCSD)

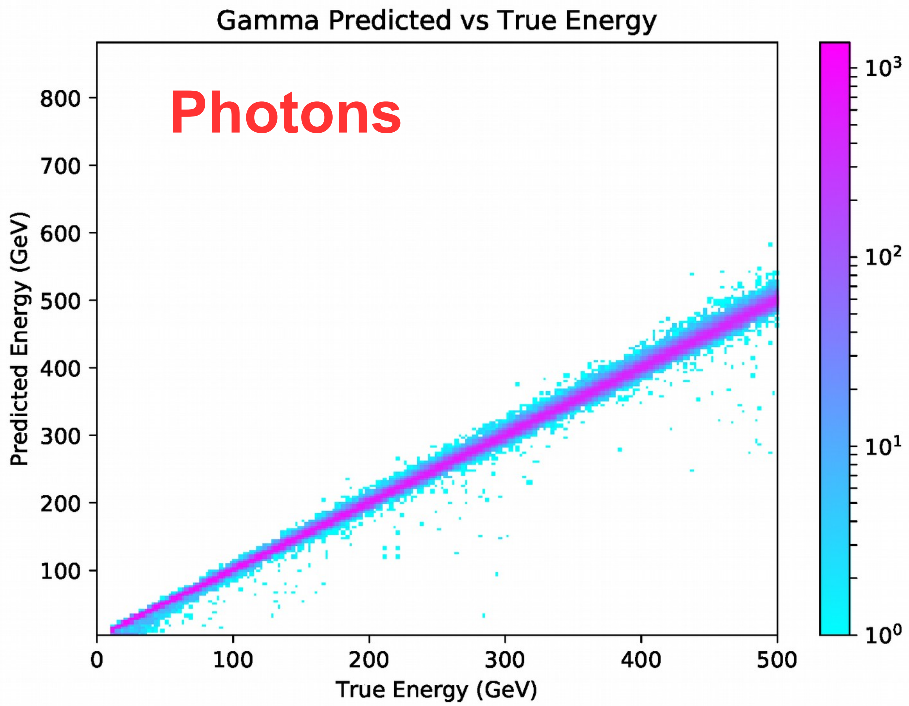
Setup

- Used some of Vitoria's code:
  - Jupyter notebook, Keras + Tensorflow backend
  - https://github.com/vitoriapacela/NotebooksLCD/blob/master/Reg_Ele_mse-checkpoint.ipynb
- CNN for energy regression, same architecture as NIPS paper (I think), details on bonus slide
- Datasets: V1, energy range 10-500 GeV, pre-split into train / val / test
  - At caltech: /bigdata/shared/LCD/V1/
- Trained on 300k electron events
  - ~20 epochs (stopped and restarted to reduce validation size)
- Evaluated for 300k electrons, photons, pi0s, charged pions
  - For electrons, results comparable to NIPS paper
    - Slightly better resolution, maybe due to more training data / epochs
  - Expect to do reasonably well for others except charged pions

Pred vs True: Electrons, Photons

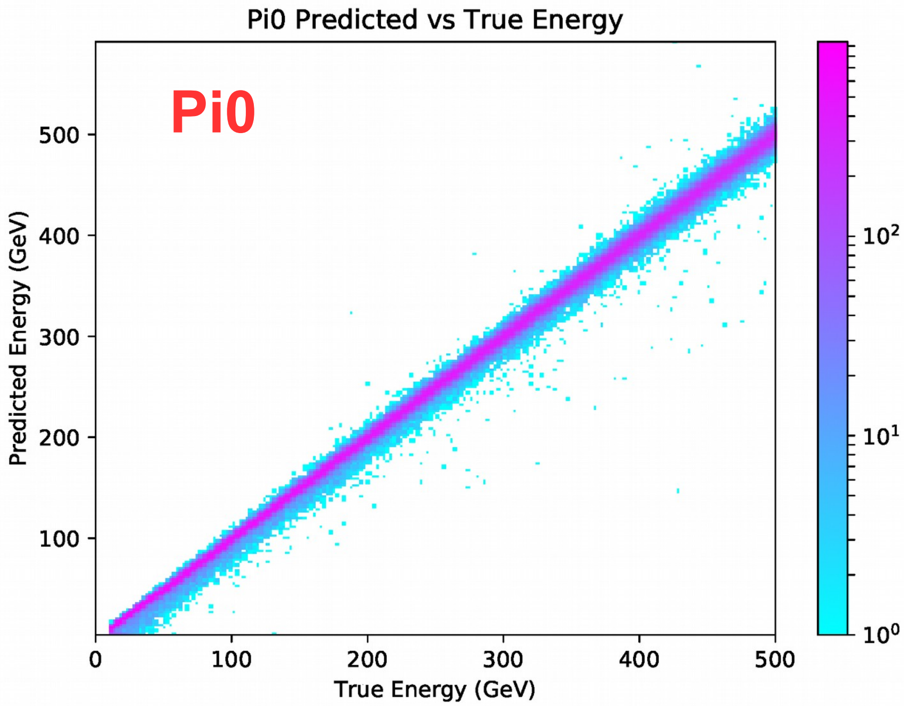
- Good agreement for electrons, used for training
- Also generally good for photons
  - Maybe a few more off-diagonal events, still small

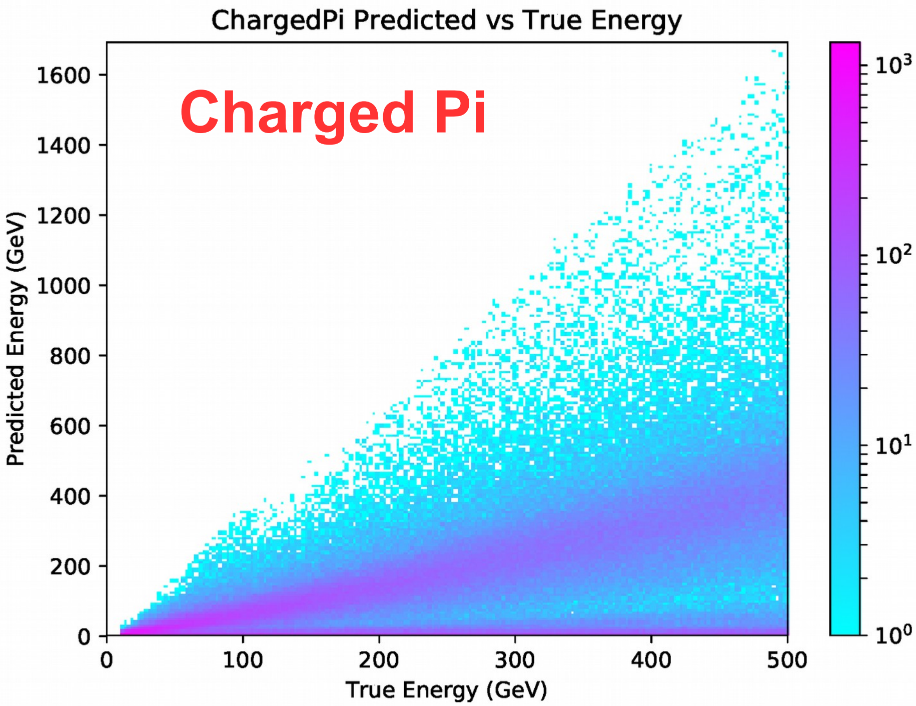




Pred vs True: Pi0, Charged Pi

- Decent agreement for Pi0, slightly wider diagonal
  - Expected since these look similar to photons
- Poor correspondence for charged pions
  - Expected since these have higher HCAL fraction
  - See a couple subpopulations, didn't look into these

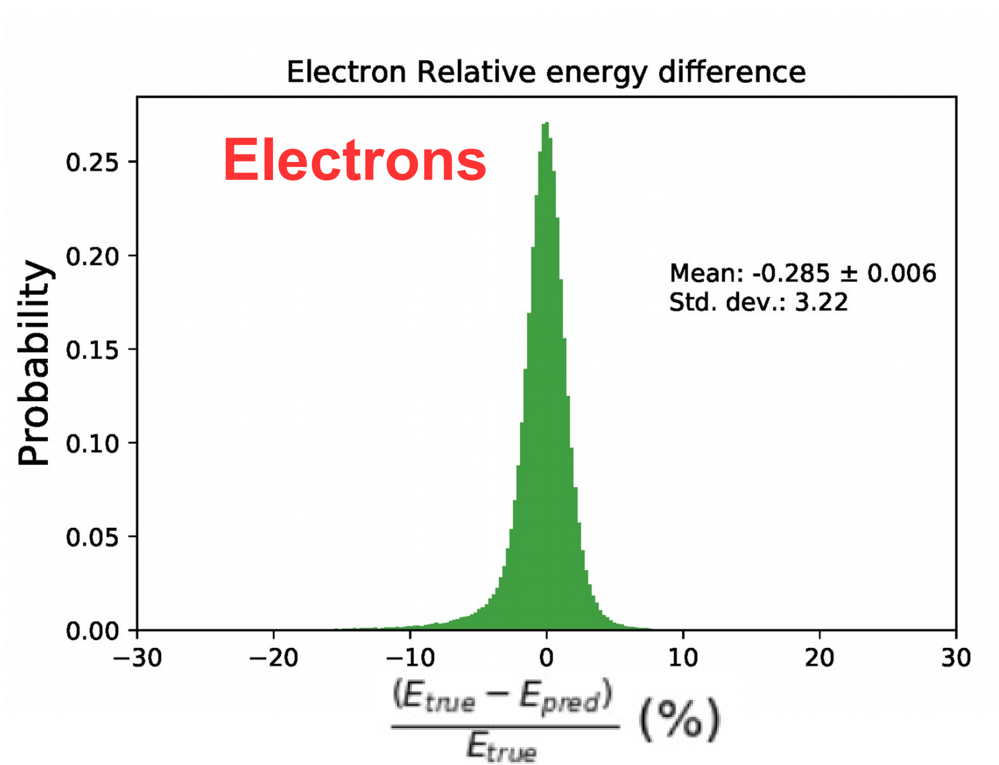


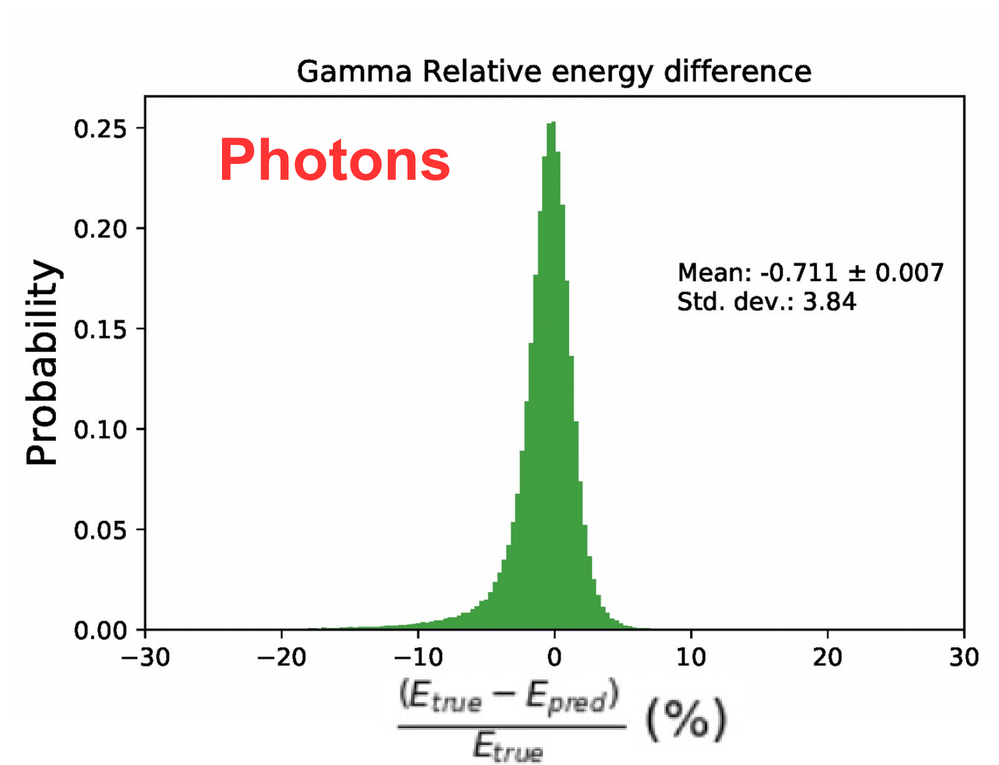


Resolution: Electrons, Photons

- Mean for electrons close to 0, sigma ~ 3.2% overall
- For photons, mean and sigma slightly worse, ~3.8%

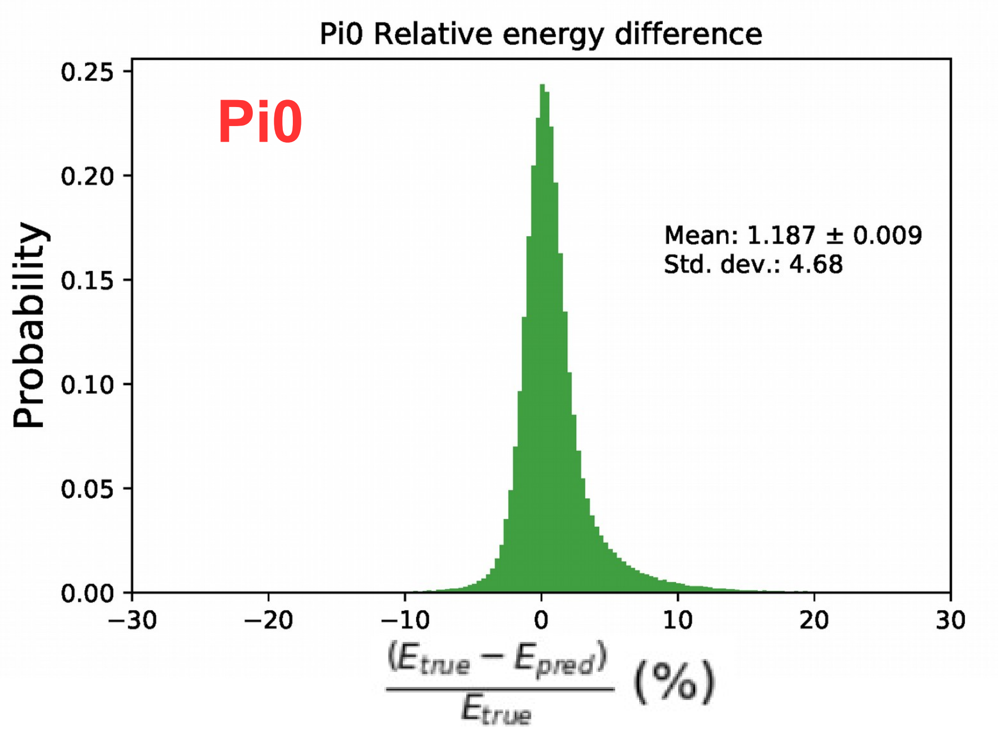
- Integrating over full energy range for these plots

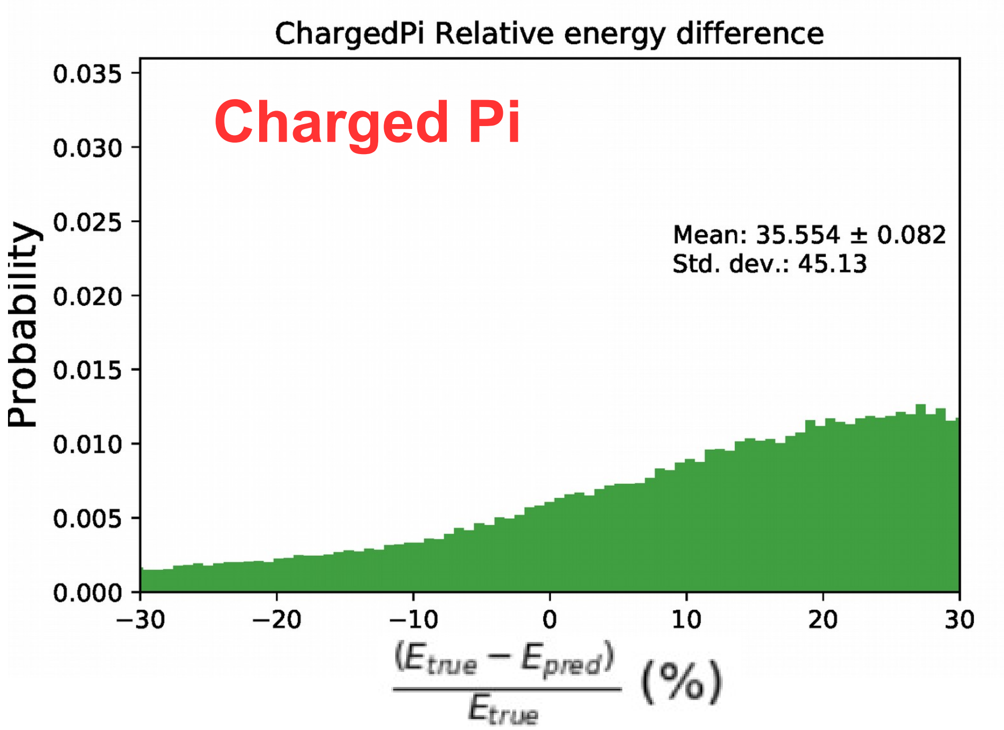




Resolution: Pi0, Charged Pi

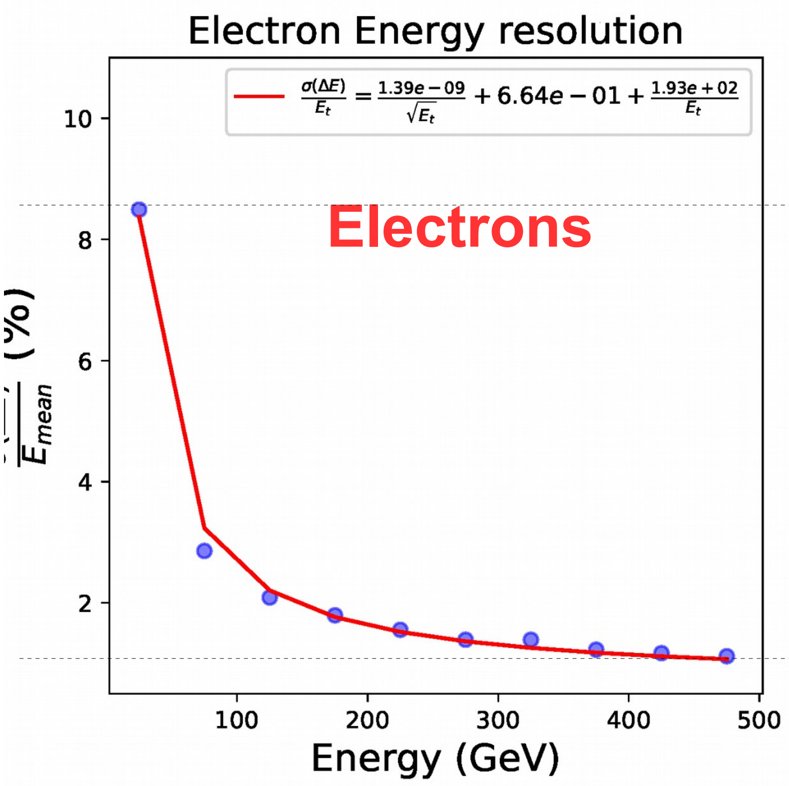
- Pi0 has a tail toward large values, underprediction of energy
  - Probably 2nd cluster not being used optimally
  - Wider sigma, ~4.7%, but affected by tail
- Charged pion has little correspondence to truth energy, as expected from 2D histogram

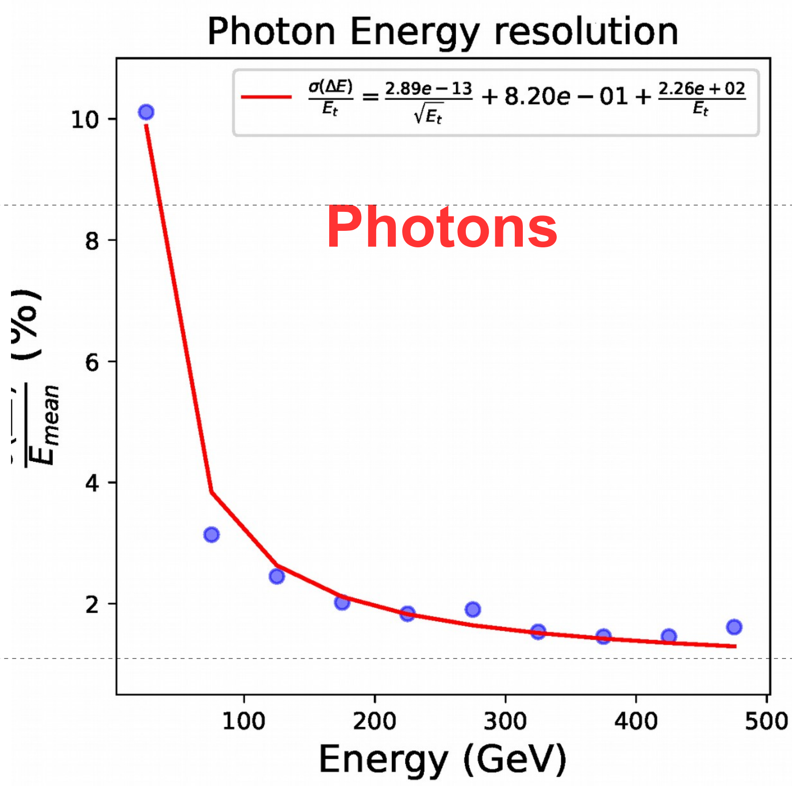




Res vs Energy: Electrons, Photons

- Electron energy resolution slightly better than photons, especially at low energy



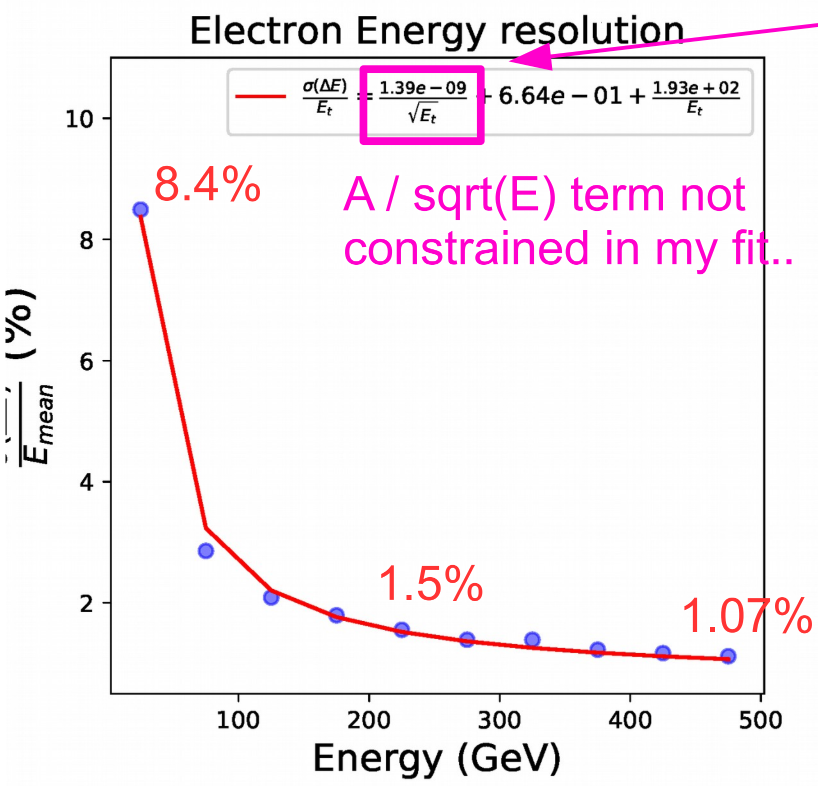


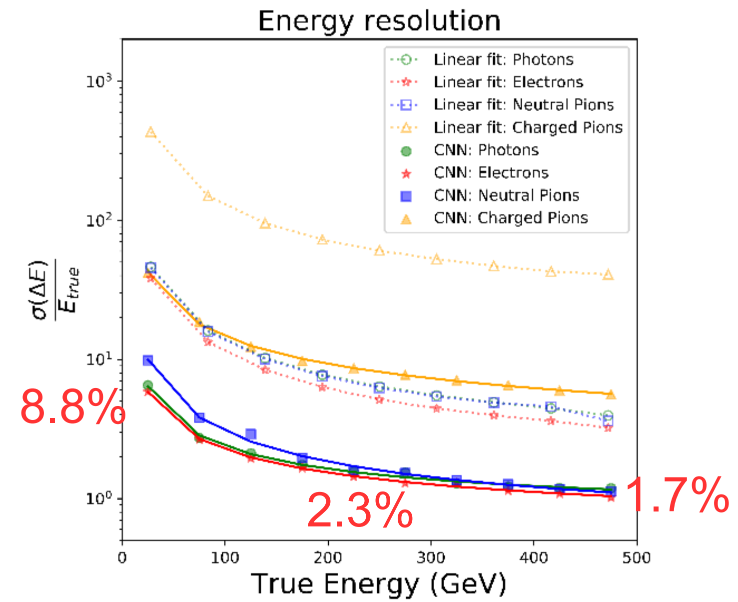
Electron Resolution vs Paper

- See better resolution than reported in NIPS paper, especially at higher energy
  - I use RMS (np.std). Did paper use gaussian fit for sigma?
  - Different sample?
  - More training events / epochs?

CNN Model			
Particle Type	a	b	c
Photons	18.3	0.75	131
Electron	$\boxed{18.7}$	0.574	111
Neutral pions	19.3	0.45	231
Charged pions	114	1.02	893

Table 2: Calorimeter resolution parameters from equation $\frac{\sigma(\Delta E)}{E_{\text{true}}} = \frac{a}{\sqrt{E_{\text{true}}}} \oplus b \oplus \frac{c}{E_{\text{true}}}$ for the resolution curves in Fig. 2.





Training Observations (1)

- Loss drops rapidly after first epoch, then improves slowly
- Training loss is reported as being ~10x larger than val_loss
  - But when I evaluated MSE manually on training data, got numbers comparable to val_loss..

```
In [*]: hist = ele_mse.fit_generator(train, samples_per_epoch=tr_samples,
                              nb_epoch=50,
                              validation_data = val,
                              nb_val_samples=10000, verbose=1,
                              callbacks=[EarlyStopping(monitor='val_loss', patience=5, verbose=1, mode='min'),
                              ModelCheckpoint(filepath='/nfshome/olivito/lcd/caloimage_2017/ele_reg_mse_{epoch:02d}.h5'
                               )
```

```
Epoch 1/50
298498/298498 [==============================] - 243s - loss: 6411.2605 - val_loss: 63.1790
Epoch 2/50
298498/298498 [==============================] - 237s - loss: 216.5278 - val_loss: 24.1511
Epoch 3/50
298498/298498 [==============================] - 241s - loss: 202.0756 - val_loss: 24.1115
Epoch 4/50
298498/298498 [==============================] - 239s - loss: 199.0847 - val_loss: 25.3108
Epoch 5/50
298498/298498 [==============================] - 242s - loss: 196.8737 - val_loss: 27.6322
Epoch 6/50
 73000/298498 [======>.......................] - ETA: 165s - loss: 199.3160- ETA: 188s
```

Training Observations (2)

- Pauses every 10k events to switch to next file
  - Using generator:
    - https://github.com/DannyWeitekamp/CMS_Deep_Learning/blob/master/CMS_Deep_Learning/io.py#L165
  - Is there a faster solution?
  - How were h5 file sizes (~150MB) chosen?

In [*]:
```
hist = ele_mse.fit_generator(train, samples_per_epoch=tr_samples,
                             nb_epoch=50,
                             validation_data = val,
                             nb_val_samples=10000, verbose=1,
                             callbacks=[EarlyStopping(monitor='val_loss', patience=5, verbose=1, mode='min'),
                             ModelCheckpoint(filepath='/nfshome/olivito/lcd/caloimage_2017/ele_reg_mse_{epoch:02d}.h5'
                              )
```

```
Epoch 1/50
298498/298498 [==============================] - 243s - loss: 6411.2605 - val_loss: 63.1790
Epoch 2/50
298498/298498 [==============================] - 237s - loss: 216.5278 - val_loss: 24.1511
Epoch 3/50
298498/298498 [==============================] - 241s - loss: 202.0756 - val_loss: 24.1115
Epoch 4/50
298498/298498 [==============================] - 239s - loss: 199.0847 - val_loss: 25.3108
Epoch 5/50
298498/298498 [==============================] - 242s - loss: 196.8737 - val_loss: 27.6322
Epoch 6/50
 73000/298498 [======>.......................] - ETA: 165s - loss: 199.3160- ETA: 188s
```

Conclusions

- Training CNN regression for one particle (electron) and applying to another works overall as expected
  - Best results for electrons, photons comparable
  - Tail of underprediction for pi0 but decent overall
  - Completely off for charged pions, as could be expected
- I see slightly better resolution for electrons than in NIPS paper
  - Not sure which differences account for this yet
- What (if any) concrete items would be interesting for update of NIPS paper?

Bonus Slides

CNN Architecture

```
# ECAL input
input1 = Input(shape=(25, 25, 25))
r = Reshape((25, 25, 25, 1))(input1)
model1 = Convolution3D(3, 4, 4, 4, activation='relu')(r)
model1 = MaxPooling3D()(model1)
model1 = Flatten()(model1)

# HCAL input
input2 = Input(shape=(5, 5, 60))
r = Reshape((5, 5, 60, 1))(input2)
model2 = Convolution3D(10, 2, 2, 6, activation='relu')(r)
#model2 = Convolution3D(10, 2, 2, 6, activation='relu')(r)
model2 = MaxPooling3D()(model2)
model2 = Flatten()(model2)

# join the two input models
bmodel = merge([model1, model2], mode='concat')  # branched model

# fully connected ending
bmodel = (Dense(1000, activation='relu'))(bmodel)
bmodel = (Dropout(0.5))(bmodel)

# oc = Dense(1,activation='sigmoid', name='particle_label')(bmodel) # output particle classification
oe = Dense(1, activation='linear', name='energy')(bmodel)  # output energy regression

# classification, will not use yet
# bimodel = Model(input=[input1,input2], output=[oc,oe])
# bimodel.compile(loss=['binary_crossentropy', 'mse'], optimizer='sgd')
# bimodel.summary()

# energy regression model
model = Model(input=[input1, input2], output=oe)
model.compile(loss=loss, optimizer='adam')
model.summary()
saveModel(model, name=name)
return model
```

CNN Summary

Layer (type)	Output Shape	Param #	Connected to
input_3 (InputLayer)	(None, 25, 25, 25)	0	
input_4 (InputLayer)	(None, 5, 5, 60)	0	
reshape_3 (Reshape)	(None, 25, 25, 25, 1)	0	input_3[0][0]
reshape_4 (Reshape)	(None, 5, 5, 60, 1)	0	input_4[0][0]
convolution3d_3 (Convolution3D)	(None, 22, 22, 22, 3)	195	reshape_3[0][0]
convolution3d_4 (Convolution3D)	(None, 4, 4, 55, 10)	250	reshape_4[0][0]
maxpooling3d_3 (MaxPooling3D)	(None, 11, 11, 11, 3)	0	convolution3d_3[0][0]
maxpooling3d_4 (MaxPooling3D)	(None, 2, 2, 27, 10)	0	convolution3d_4[0][0]
flatten_3 (Flatten)	(None, 3993)	0	maxpooling3d_3[0][0]
flatten_4 (Flatten)	(None, 1080)	0	maxpooling3d_4[0][0]
merge_2 (Merge)	(None, 5073)	0	flatten_3[0][0] flatten_4[0][0]
dense_2 (Dense)	(None, 1000)	5074000	merge_2[0][0]
dropout_2 (Dropout)	(None, 1000)	0	dense_2[0][0]
energy (Dense)	(None, 1)	1001	dropout_2[0][0]

Total params: 5,075,446
Trainable params: 5,075,446
Non-trainable params: 0

~5M parameters