Final Project

Handwriting Recognition for Mathematical Texts

Beerepoot, Erik

CSCI S-89a Deep Learning, Summer 2019

Harvard University Extension School

Prof. Zoran B. Djordjević

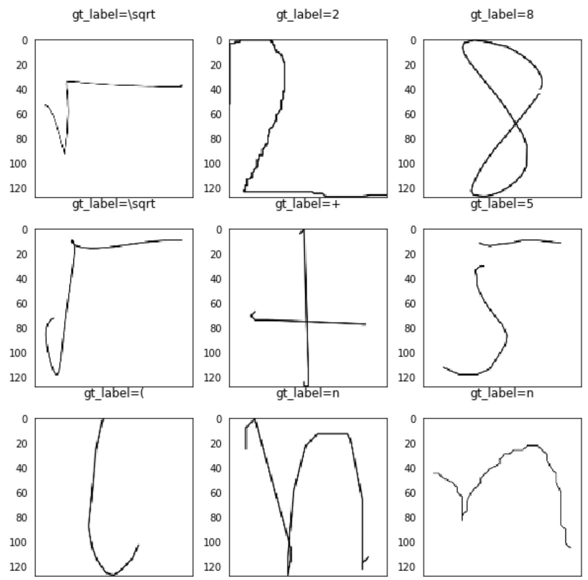
Introduction

- Problem:
  - Typesetting mathematics is an error prone and laborious process.
  - Quicker & more natural to write by hand.
  - Reproducing mathematics from papers suffers from similar problems.

- Solution:
  - Train a neural network to recognize mathematical formulas and generate the corresponding Latex.

Datasets

- A few datasets exist:
  - Competition on Recognition of Online Handwritten Mathematical Expressions (CROHME)
    - 6 years worth of data.
    - Mostly small amounts of data — largest aggregated dataset is ~80k symbols, others ~10k
    - InkML format
  - Harvard Im2latex dataset
    - ~85k training formulas
    - ~10k test & validation formulas
  - InftyCDB
    - Typeset symbols with various distortions
    - ~100k examples
  - HASYv2
    - ~170k handwritten symbols
- Using im2latex for formula recognition and CROHME



Samples from CROHME dataset

Approach

- Two prong approach

- First, train a CNN to predict symbols only
  - Simpler problem means we can iterate quickly
  - Explore encoder architecture for use in more complex model

- Second, train a seq2seq model to generate Latex from images
  - More complex
  - Slow to train
  - Use encoder model from prior work

Symbol Recognition using CNN - Data Preprocessing

Notebook 1.1

- Convert the data from stroke-based format (InkML) to .png.
- Perform a visual inspection of data.
- Pickle the result for easy loading using Pandas.

Sample images generated from CROHME data

label = “(“ label = "S” label = “=“

Symbol Recognition using CNN - The Model

Notebook 1.2

- Train 3 models:
  - Very simple dense network for binary prediction
    - Baseline result
  - Simple CNN for binary label prediction
    - Baseline for CNN
    - Performs better!
  - More complex CNN for k-class symbol prediction
    - MNIST on Steroids!
    - Surprisingly effective, given enough data.(high 90s acc.)

```
# Add CNNs for features
k_conv_model = models.Sequential()
k_conv_model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(size[0], size[1], 1)))
k_conv_model.add(layers.MaxPooling2D((2, 2)))
k_conv_model.add(layers.Conv2D(64, (3, 3), activation='relu'))
k_conv_model.add(layers.MaxPooling2D((2, 2)))
k_conv_model.add(layers.Conv2D(64, (3, 3), activation='relu'))

# Add dense layers for classification
k_conv_model.add(layers.Flatten(input_shape=(size[0], size[1], 1)))
k_conv_model.add(layers.Dense(25, activation='relu'))
k_conv_model.add(layers.Dense(len(unique_labels), activation='softmax'))

k_conv_model.compile(optimizer=tf.optimizers.Adam(learning_rate),
              loss=tf.losses.SparseCategoricalCrossentropy(),
              metrics=[metrics.sparse_categorical_accuracy])

k_conv_model.summary()
```

Formula recognition using seq2seq - Data Preprocessing

Notebook 2.1

- Render ~100k examples in Latex to pngs.
- Cropped & grayscale.
- Normalize latex
  - Different expressions can be rendered identically!

$\alpha + \beta$ ⟷ \alpha + \beta

$\frac{1}{1+e^{-x}}$ ⟷ \frac { 1 } { 1 + e ^ { - x } }

$G_{\mu v} = 8\pi G(T_{\mu v} + \rho_v g_{\mu v})$ ⟷ G _ { \mu v } = 8 \pi G (T _ { \mu v } + \rho _ { v } g _ { \mu v })

The model(s) - Formula Prediction

Notebook 2.2

- Seq2Seq model with Bahdanau attention
- CNN Encoder + linear “embedding layer”
- Attention over image features
- Concat attention vector with input token
- Pass through GRU to generate output

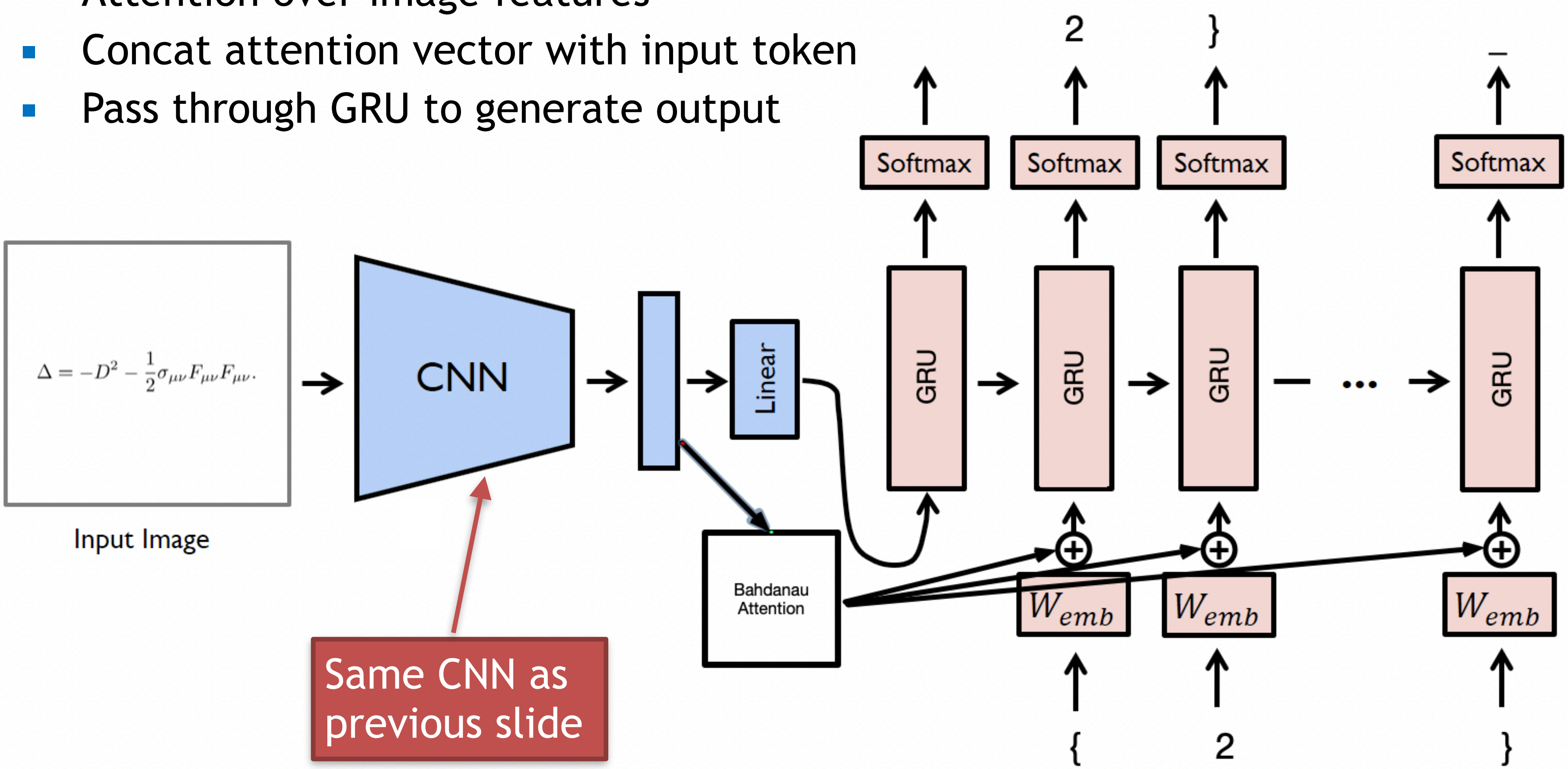
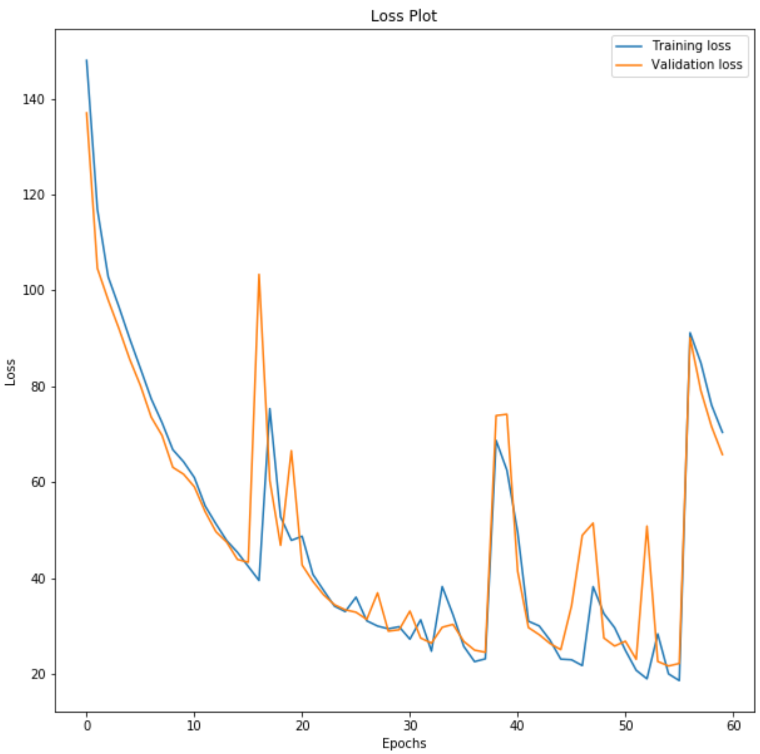


Diagram adapted from: https://www.analyticsvidhya.com/blog/2018/04/solving-an-image-captioning-task-using-deep-learning/

Training the model

Notebook 1.2 & 2.2



- Training is noisy!
  - Sudden peaks in the loss value.
  - Likely indicates numerical instability in the optimizer.
- Training is slow
  - Took ~9 hours on AWS p3.8xlarge.
  - Using 4 Tesla V100 GPUs.
  - With ~50% of full dataset (~32k examples).
- Convergence is not guaranteed
  - Spent significant time exploring encoder architectures – simpler is faster & better!

Results - Accurate predictions

Original	Predicted
$j_1^k = \omega_1^{k-2} \subseteq \omega_1^k$	$j_1^k = \omega_1^{k-2} \subset \omega_1^k$
$\Delta = -D^2 - \frac{\boxed{i}}{2}\sigma_{\mu\nu}F_{\mu\nu}.$	$\Delta = -D^2 - \frac{1}{2}\sigma_{\mu\nu}F_{\mu\nu}F_{\mu\nu}.$
$\tau_j = 1 - yexp(i(\mu_a\boxed{(2j) - \varepsilon)}).$	$\tau_j = 1 - yexp(i(\mu_a(2(\mu j)).$

Results - Poor predictions

Original

$$\mathcal{U}'' + \frac{2}{r}\mathcal{U}' - (\mathcal{U}')^2 = \frac{1}{4}\left(h_1'\right)^2 + \frac{1}{4}\left(h_2'\right)^2 ,$$

Predicted

Long sequences are hard!

$$\mathcal{U}'' + \frac{2}{r}\mathcal{J}' - (\mathcal{U}')^{\in} \boxed{(\langle_{\in}^{\in} (\langle_{\in}^{\in} (\langle_{\in}^{\in} (\langle_{\in}^{\in} (\langle_{\in}^{\in} (\langle_{\in}^{\in}}$$

Results - Poor predictions

Original

$$\int_{-\infty}^{\infty} \frac{dx}{(\gamma + x^2)(\delta + x^2)} = \frac{\pi}{\sqrt{\gamma\delta}(\sqrt{\gamma} + \sqrt{\delta})} = \frac{\pi}{\sqrt{\gamma\delta}} \frac{\sqrt{\delta} - \sqrt{\gamma}}{\delta - \gamma},$$

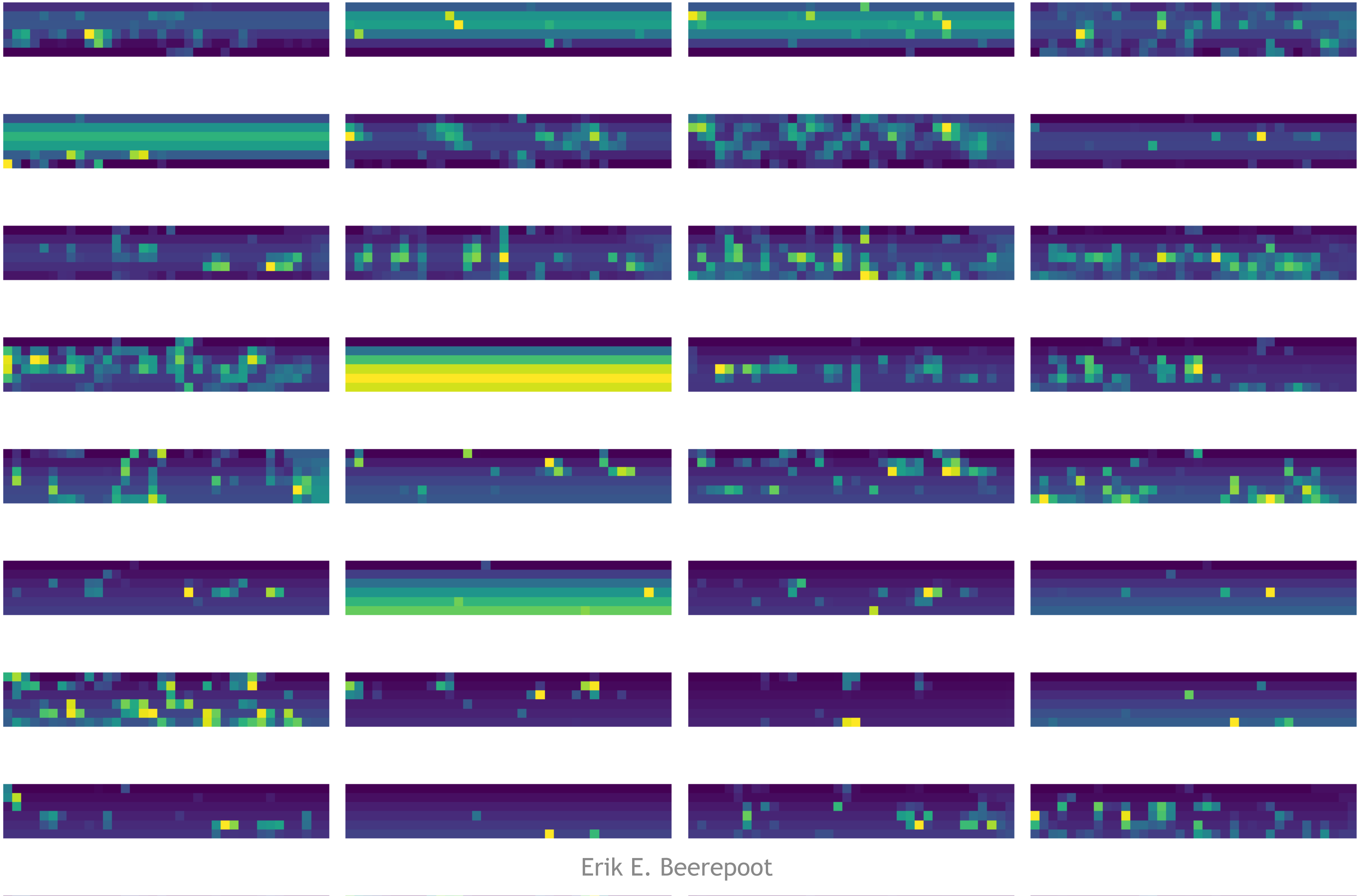
Predicted

Repeated tokens for long sequences

$$\int_{-\infty} \frac{dx}{\delta^2)} = \boxed{\frac{\pi}{\sqrt{\gamma}} \frac{\pi}{\sqrt{\gamma}} \frac{\pi}{\sqrt{\gamma}} \frac{\pi}{\sqrt{\gamma}} \frac{\pi}{\sqrt{\gamma}} \frac{\pi}{\sqrt{\gamma}} \frac{\pi}{\sqrt{\gamma}}}$$

CNN Activations

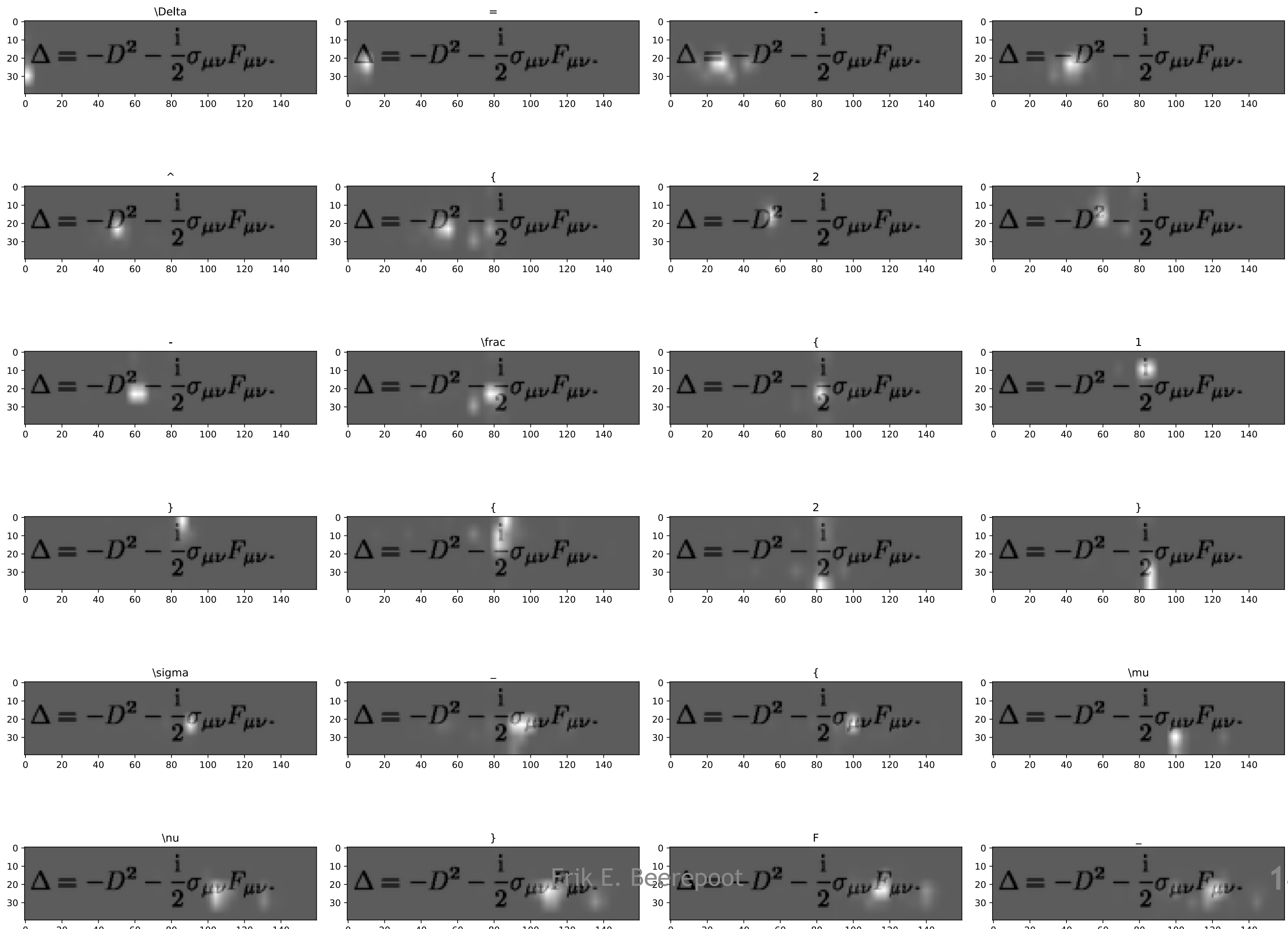
CNN Activations for $\Delta = -D^2 - \frac{1}{2}\sigma_{\mu\nu}F_{\mu\nu}F_{\mu\nu}$.



Erik E. Beerepoot

Attention plot

$$\Delta = -D^2 - \frac{1}{2}\sigma_{\mu\nu}F_{\mu\nu}F_{\mu\nu}.$$



Successes & Challenges

- ✅ Successes
  - Seq2Seq Model training was successful with a subset of the data.
  - Subjectively, results were quite reasonable.
  - Great springboard for future experimentation.
  - Learned a lot!
- ❌ Challenges
  - Hard to get model to converge — attention was required!
  - Many small mistakes meant hours spent debugging.
  - Training is slow and/or costly.
  - Distributed training was trickier than anticipated.

Conclusion

- **Handwriting Recognition for Mathematical Texts**

1. Erik Beerepoot

- **Two minute (short) video:**
  - https://youtu.be/Fwr0lHJuzDI
- **Reference Links:**
  - https://github.com/erikbeerepoot/img-to-latex