PROJECT REPORT

HANDWRITTEN MATHEMATICAL EXPRESSION RECOGNITION

STATISTICAL METHODS IN AI

Aditi Shrivastava (2018201056)
Bhavi Dhingra (2018201058)
Hitesh Kaushik (2018201057)
Ravi Jakhania (2018201018)

Contents

0.1 Problem Description

Handwritten mathematical expression recognition is an appealing topic in pattern recognition field since it exhibits a big research challenge and underpins many practical applications. From a scientific point of view, a large set of symbols (more than 100) needs to be recognized, and also the 2 dimensional (2-D) structures (specifically the relationships between a pair of symbols, for example superscript and subscript), both of which increase the difficulty of this recognition problem.

Mathematical expressions are indispensable for describing problems in maths, physics and many other fields. Meanwhile, people have begun to use handwritten mathematical expressions as one natural input mode. However, machine recognition of these handwritten mathematical expressions is difficult and exhibits three distinct challenges, i.e., the complicated two-dimensional structures, enormous ambiguities coming from handwriting input and variant scales of handwritten math symbols.

Handwritten mathematical expression recognition comprises two major problems: symbol recognition and structural analysis. The two problems can be solved sequentially or globally. However, both conventional sequential and global approaches have the following limitations:
1) The challenging symbol segmentation is inevitable, which brings many difficulties;
2) The structural analysis is commonly based on two dimensional context free grammar, which requires prior knowledge to define a math grammar;
3) The complexity of parsing algorithms increases with the size of math grammar.

0.2 Overview Of Proposed Solution

In recent research of deep learning, a novel attention based encoder-decoder model has been proposed. Its general application in machine translation, speech recognition, character recognition and image captioning inspires researchers that mathematical expression recognition can also be one proper application. More specifically, a recent paper proposed a model namely WAP. The WAP learns to encode input expression images and decode them into LATEX strings.

The encoder is a convolutional neural network (CNN) based on VGG architecture that maps images to high-level features. The decoder is a recurrent neural network (RNN) with gated recurrent units (GRU) that converts these high-level features into output strings one symbol at a time. For each predicted symbol, an attention model built in the decoder scans the entire input expression image and chooses the most relevant region to describe a math symbol or an implicit spatial operator. Compared with conventional approaches for handwritten mathematical expression recognition, the attention based encoder-decoder model possesses three distinctive properties:
1) It is end-to-end trainable.
2) It is data-driven, in contrast to traditional systems that require a predefined math grammar.
3) Symbol segmentation can be automatically performed through attention model.

In this paper, the authors still focus on offline handwritten mathematical expression recognition and report recent progress on WAP model. The main contribution is in two aspects. Firstly, they improve the CNN encoder by employing a novel architecture called densely connected convolutional networks **(DenseNet)**. The DenseNet has shown excellent performance on image classification task as it strengthens feature extraction and facilitates gradient propagation. Secondly, they present a novel **multi-scale attention model** to deal with the problems caused by pooling operations. Although pooling layers are essential parts of convolutional networks, they shrink the size of feature maps, yielding decrease of resolution. Because the scales of handwritten math symbols vary severely, the fine-grained details of extracted feature maps are especially important in handwritten mathematical expression recognition, which are lost in low-resolution feature maps.

0.3 Literature Survey

Both symbol recognition and structure analysis of two dimensional patterns have been extensively studied for decades. Mathematical expression recognition, which features both of them as the two major stages of the recognition process, is a good subject for studying the integration of the two areas. Many symbol recognition techniques work under the assumption that the symbols have already been isolated from each other.

Since last four decades, the research has been done on handwritten numeric data recognition (G.G.Rajput and S.M.Mali, (2010); V. Vijaya Kumar et al., 2010; M. Karic and G.Martivonic (2013)) and mathematical symbol recognition (Birendra Keshari and Stephen M. Watt, 2008; Xie Xiaofang, 2008; Christopher Malon et al., 2008; Francisco Alvaro and John Andren Sanchez, 2010; Dipak D.Bage et al., 2013). Most of the research work considered simple mathematical expressions, works with binary operators, integration and trigonometric equations (His-Jian Lee and J. Wang, 1995; Kazuki Ashida et al., 2006; Sanjay S. Gharde et al., 2012). Most of the research work was reported on Online handwritten mathematical symbol recognition (Hans Jurgen Winkler and Manfred Lang,1997; Utpal Garain et al., 2004; Birendra Keshari and Stephen M. Watt, 2008; Dipak D.Bage et al., 2013). Considerable research is still needed in the off-line printed and handwritten mathematical symbol recognition. Research into recognizing handwritten mathematics began in the late 1960's. The rate of progress was limited in this problem, with absence of benchmark datasets and standard evaluation tools. In late 1960s, the problem of recognizing handwritten ME was originated with Anderson's research using structural analysis approach.

Francisco Alvaro et al. (2010) had compared the classification techniques for offline printed mathematical symbol recognition on two databases. The proposed techniques were evaluated in two different databases: the UWIII database that was a small database with degraded images and the InftyCDB-1 database that was a large database with good quality images. Three classification methods HMM, KNN and SVM were compared.

Richard Zanibbi et al. (2002) had proposed efficient system for recognizing typeset and handwritten mathematical notations. This paper described the design and implementation of a mathematics recognition system that makes extensive use of tree transformation. The baseline tree was built, then lexical pass was used to group labels and symbols, further expression syntax was checked by using operator precedence and associativity. This paper dealt with structural analysis of mathematical expressions. Ernesto Tapia et al. (2004) presented a structural analysis method for the recognition of on-line handwritten mathematical expressions based on a minimum spanning tree construction and symbol dominance (superscripts and subscripts). Taik Heon Rhee et al. (2009) proposed a layered search tree framework for online handwritten ME recognition by using structural analysis. ME structures were expanded by adding symbol hypotheses using handwritten strokes. For expansion BFS (Breadth First Search) was used and the spatial relationship was used to create numbers of branches in the expansion of an expression. Kang Kim et al. (2009) presented a rule-based approach that utilizes some types of contextual information to improve the accuracy of handwritten mathematical expression (ME) recognition. The base system used for evaluation was a handwritten ME recognizer developed by Rhee et al. lysis.

Given the two-dimensional structure of mathematical notation, most approaches are based on predefined grammars as a natural way to solve the problem. In fact, the first proposals on ME recognition belonged to this case. Subsequently, different types of grammars have been investigated. For example, Chan and Yeung used definite clause grammars, the Lavirotte and Pottier model was based on graph grammars, Yamamoto et al. Another paper presented a system using Probabilistic Context-Free Grammars (PCFG), and MacLean and Labahn developed an approach using relational grammars and fuzzy sets.

Proposals based on PCFG use a probability model to analyse the structure of the expression and address ambiguities in handwritten data. Alvaro and Sanchez proposed a system that parses expressions using two-dimensional stochastic context-free grammars. This is a global approach that combines several stochastic sources of information to globally determine the most likely expression. Yamamoto et al. presented a version of the CYK (Cocke-Younger-Kasami) algorithm for parsing two-dimensional PCFG (2D-PCFG). They defined probability functions based on a region representation called the "hidden writing area". The model proposed by Awal et al. considers several segmentation hypotheses based on spatial information, and the symbol classifier includes a rejection

class, namely, "junk", to avoid incorrect segmentations and provide useful information to structure the analyser. MacLean and Labahn presented a Bayesian model for recognizing HME; this model captures all recognizable interpretations of the input and organises them in a parse forest.

Recently, a novel neural network model, namely encoder-decoder, has been exploited specifically to address sequence to sequence learning. The encoder-decoder is designed to handle variable-length input and output sequences. Typically, both the encoder and the decoder are recurrent neural networks (RNN). The encoder RNN learns to encode the sequential variable-length input into a fixed-length vector. The decoder RNN then uses this vector to produce the variable-length output sequence, one word at a time. In the training stage, to generate the next predicted word, the model provides the ground truth labels from previous words as inputs to the decoder. In the inference stage, the model utilises a beam search to obtain suitable candidates for the next predicted word. Such a framework has been applied extensively to many applications including machine translation, parsing and speech recognition.

Harold Mouchere et al. (2013) had reported on the third international Competition on Handwritten Mathematical Expression Recognition (CROHME), in which eight teams from academia and industry took part. Training dataset was expanded to over 8000 expressions, and new tools were developed for evaluating performance at the level of strokes as well as online expressions and symbols.

0.4 Dataset

The CROHME 2014 dataset had a train set of 8836 math ex- pressions (86K symbols) and a test set of 986 math expressions (6K symbols). There were 101 math symbol classes. None of the hand-written expressions or LaTeX notations in the test set appeared in the train set. Following other participants in CROHME 2014, we used the CROHME 2013 test set as a validation set for estimating the models during the training.
The handwritten strokes are recorded as inkml format data shown below :

```
<ink xmlns="http://www.w3.org/2003/InkML">
<traceFormat>
<channel name="X" type="decimal"/>
<channel name="Y" type="decimal"/>
</traceFormat>
<annotation type="UI">2011_IVC_DEPART_F002_E018</annotation>
<annotation type="writer">depart002</annotation>
<annotation type="truth">$\sin (nx)$</annotation>
<annotation type="age">26</annotation>
<annotation type="gender">M</annotation>
<annotation type="hand">L</annotation>
<annotation type="copyright">LUNAM/IRCCyN</annotation>
<annotationXML type="truth" encoding="Content-MathML">
       <math xmlns='http://www.w3.org/1998/Math/MathML'>
              <mrow>
                     <mi xml:id="sin_1">\sin</mi>
                     <mrow>
                            <mo xml:id="(_1">(</mo>
                            <mrow>
                                   <mi xml:id="n_1">n</mi>
                                   <mrow>
                                          <mi xml:id="x_1">x</mi>
                                          <mo xml:id=")_1">)</mo>
                                   </mrow>
                            </mrow>
                     </mrow>
              </mrow>
       </math>
</annotationXML>
<trace id="0">
11.3393 25.5279, 11.3393 25.5279, 11.3313 25.5279, 11.3112 25.544, 11.2711 25.5761, 11.2149 25.6162, 11.1788 25.6563, 11.1507 25.6764,
11.1467 25.6804, 11.1507 25.6804, 11.1587 25.6804, 11.1788 25.6844, 11.2109 25.7005, 11.251 25.7205, 11.2912 25.7446, 11.3072 25.7647,
11.3112 25.7887, 11.3152 25.8208, 11.2912 25.8449, 11.2631 25.865, 11.231 25.877, 11.2149 25.881, 11.2029 25.877, 11.1989 25.873, 11.1989
25.869
</trace>
<trace id="1">
11.4677 25.7285, 11.4677 25.7285, 11.4637 25.7406, 11.4637 25.7606, 11.4517 25.7807, 11.4396 25.8048, 11.4316 25.8369, 11.4236 25.861
</trace>
```

Figure 1: Inkml file format

0.5 Problem Design

In this section, we first make a brief summarization of DenseNet since the encoder is based on densely connected convolutional blocks. Then we introduce the classic attention based encoder-decoder framework. Finally, we extend DenseNet by introducing a multi-scale dense encoder and describe the implementation of multi-scale attention model in detail.

1. Dense Encoder
   The main idea of DenseNet is to use the concatenation of output feature maps of preceding layers as the input of succeeding layers. As DenseNet is composed of many convolution layers, let $H_l(.)$ denote the convolution function of the l^{th} layer, then the output of layer l is represented as:

$$x_l = H_l([x_0; x_1; ...; x_{l-1}) \tag{1}$$

   where $x_0, x_1, ..., x_l$ denote the output features produced in layers 0, 1, . . . , l, ";" denotes the concatenation operation of feature maps. This iterative connection enables the network to learn shorter interactions cross different layers and reuses features computed in preceding layers. By doing so the DenseNet strengthens feature extraction and facilitates gradient propagation.

   An essential part of convolutional networks is pooling layers, which is capable of increasing receptive field and improving invariance. However, the pooling layers disenable the concatenation operation as the size of feature maps changes. Also, DenseNet is inherently memory demanding because the number of inter-layer connections grows quadratically with depth. Consequently, the DenseNet is divided into multiple densely connected blocks as shown in Fig. 2. A compression layer is appended before each pooling layer to further improve model compactness.

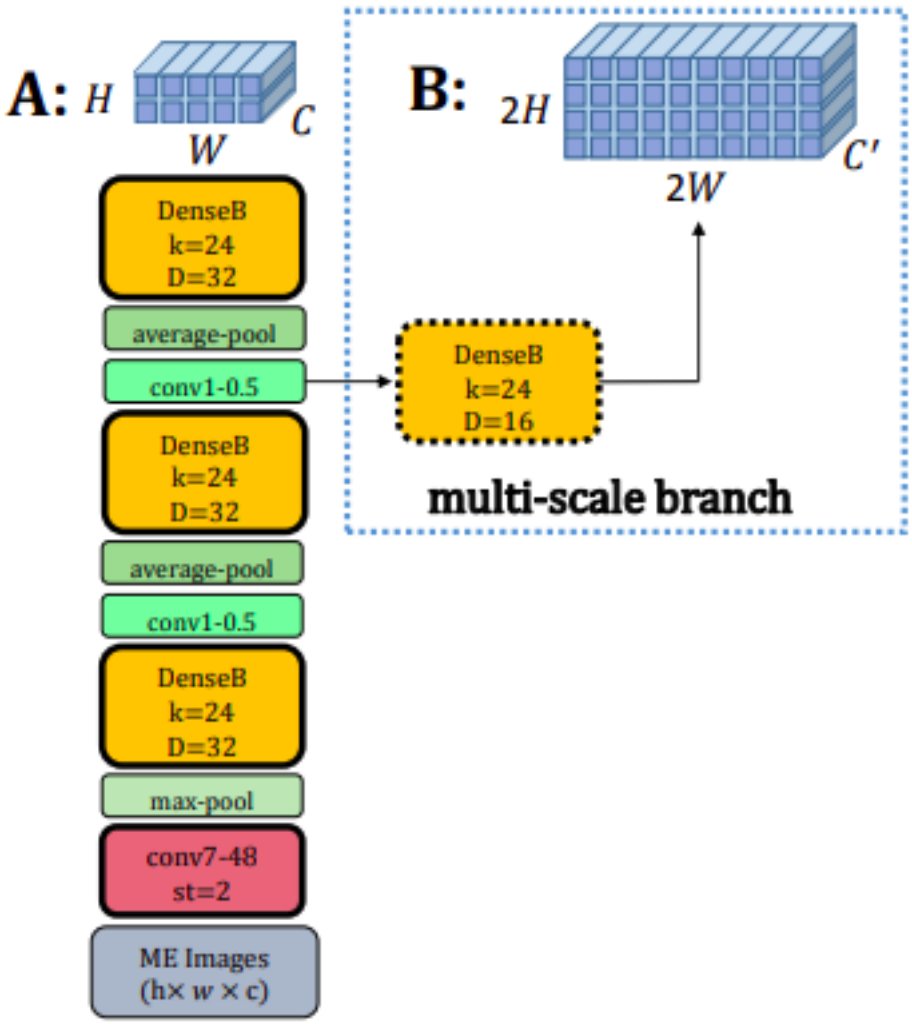

Figure 2: Architecture of multi-scale dense encoder. The left part is the main branch while the right part is the multi-scale branch.

2. Decoder

   We employ GRU as the decoder because it is an improved version of simple RNN which can alleviate the vanishing and exploding gradient problems. Given input x_t, the GRU output h_t is computed by:

$$h_t = GRU(x_t, h_{t-1}) \tag{2}$$

   and the GRU function can be expanded as follows:

$$z_t = \sigma(W_{xz}x_t + U_{hz}h_{t-1}) \tag{3}$$

$$r_t = \sigma(W_{xr}x_t + U_{hr}h_{t-1}) \tag{4}$$

$$\tilde{h}_t = \tanh(W_{xh}x_t + U_{rh}(r_t \otimes h_{t-1})) \tag{5}$$

$$h_t = (1 - z_t) \otimes h_{t-1} + z_t \otimes \tilde{h}_t \tag{6}$$

where σ is the sigmoid function and $\otimes$ is an element-wise multiplication operator. z_t, r_t *and* $\tilde{h}_t$ are the update gate, reset gate and candidate activation, respectively. Assuming the output of CNN encoder is a three-dimensional array of size $HxWxC$, consider the output as a variable length grid of L elements, $L = HxW$. Each of these elements is a C-dimensional annotation that corresponds to a local region of the image.

$$A = \{a_1, ..., a_L\}\ ,\ a_i \in R^C \tag{7}$$

Meanwhile, the GRU decoder is employed to generate a corresponding LaTeX string of the input mathematical expression. The output string Y is represented by a sequence of one-hot encoded symbols.

$$Y = \{y_1, ..., y_T\}\ ,\ y_i \in R^K \tag{8}$$

where K is the number of total symbols in the vocabulary and T is the length of LaTeX string. Note that, both the annotation sequence A and the LaTeX string Y are not fixed-length. To address the learning problem of variable-length annotation sequences and associate them with variable-length output sequences, they attempt to compute an intermediate fixed-length vector c_t, namely context vector, at each decoding step t. The context vector c_t is computed via weighted summing the variable-length annotations a_i:

$$c_t = \sum_{i=1}^{L} \alpha_{ti} a_i \tag{9}$$

Here, the weighting coefficients α_{ti} are called attention probabilities and they will make decoder to know which part of input image is the suitable place to attend to generate the next predicted symbol and then assign a higher weight to the corresponding local annotation vectors a_i. After computing the intermediate fixed-length context vector, we then generate the LaTeX string one symbol at a time. By doing so, the problem of associating variable-length annotation sequences with variable-length output LaTeX strings is addressed.

The probability of each predicted symbol is computed by the context vector c_t, current decoder state s_t and previous target symbol y_{t-1} using the following equation:

$$p(y_t|y_{t-1}, X) = g(W_o h(Ey_{t-1} + W_s s_t + W_c c_t)) \tag{10}$$

where X denotes input mathematical expression images, g denotes a softmax activation function, h denotes a maxout activation function, let m and n denote the dimensions of embedding and GRU decoder state respectively, then $W_o \in R^{K \times \frac{m}{2}}$ and $W_s \in R^{m \times n}$, E denotes the embedding matrix.

3. Multi-Scale Attention with Dense Encoder

   (a) Multi-Scale Dense Encoder: To implement the multiscale attention model, the authors first extend the single-scale dense encoder into multi-scale dense encoder. As illustrated in Fig. 2, the dense encoder consists of two branches, i.e., except the main branch which produces low-resolution annotations A, our dense encoder has another multi-scale branch that produces high-resolution annotations B. The multi-scale branch is extended before the last pooling layer of the main branch so that the output feature maps of multi-scale branch has a higher resolution. The high-resolution annotation is a three dimensional array of size $2H \times 2W \times C'$, which can be represented as a variable-length grid of 4L elements:

$$B = b_1, ..., b_{4L}, b_i \in R^{C'} \tag{11}$$

   where L is the length of annotation sequence A. Intuitively, they can extend several multi-scale branches before every pooling layer but such operation brings too much computational cost as the size of feature maps becomes too large.

   As for the implementation details of dense encoder, they employ three dense blocks in the main branch as described by yellow rectangles in Fig. 2. Before entering the first dense block, a 7 × 7 convolution (stride is 2 × 2) with 48 output channels is performed on the input expression images, followed by a 2 × 2 max pooling layer. Each dense block is titled as "DenseB" because they use bottleneck layers to improve computational efficiency, i.e. a 1 × 1 convolution is introduced before each 3 × 3 convolution to reduce the input to 4k feature maps. The growth rate k = 24 and the depth (number of convolution layers)

of each block D = 32 which means each block has 16 1 × 1 convolution layers and 16 3 × 3 convolution layers. A batch normalization layer and a ReLU activation layer are performed after each convolution layer consecutively. We use 1 × 1 convolution followed by 2 × 2 average pooling as transition layers between two contiguous dense blocks. The transition layer reduces the number of feature maps of each block by half. While in the multi-scale branch, they append another dense block with bottleneck layer, k = 24 and D = 16.

(b) Multi-Scale Attention Model: In this study, the decoder adopts two unidirectional GRU layers to calculate the decoder state s_t and the multi-scale context vector c_t that are both used as input to calculate the probability of predicted symbol in Eq. (10). The authors employ two different single-scale coverage based attention model to generate the low-resolution context vector and high-resolution context vector by attending to low-resolution annotations and high-resolution annotations respectively. As the low-resolution context vector and high-resolution context vector have the same length 1, they concatenate them to produce the multi-scale context vector:

$$\hat{s}_t = GRU(y_{t-1}, s_{t-1}) \tag{12}$$

$$cA_t = f_{catt}(A, \hat{s}_t) \tag{13}$$

$$cB_t = f_{catt}(B, \hat{s}_t) \tag{14}$$

$$c_t = [cA_t; cB_t] \tag{15}$$

$$s_t = GRU(c_t, \hat{s}_t) \tag{16}$$

where s_{t-1} denotes the previous decoder state, $\hat{s}_t$ is the prediction of current decoder state, cA_t is the low-resolution context vector at decoding step t, similarly cB_t is the highresolution context vector. The multi-scale context vector ct is the concatenation of cA_t and cB_t and it performs as the input during the computation of current decoder state s_t.

f_{catt} denotes a single-scale coverage based attention model. Take the computation of low-resolution context vector cA_t as an example, we parameterize f_{catt} as a multi-layer perceptron:

$$F = Q * \sum_{l=1}^{t-1} \alpha_l \tag{17}$$

$$e_{ti} = \nu_{att}^{T} \ tanh(U_s \hat{s}_t + U_a a_i + U_f f_i) \tag{18}$$

$$\alpha_{ti} = \frac{exp(e_{ti})}{\sum_{k=1}^{L} exp(e_{tk})} \tag{19}$$

$$cA_t = \sum_{i=1}^{L} \alpha_{ti} a_i \tag{20}$$

where a_i denotes the element of low-resolution annotation sequence A, e_{ti} denotes the energy of a_i at time step t conditioned on the prediction of current decoder state $\hat{s}_t$ and coverage vector f_i . The coverage vector is initialized as a zero vector and they compute it based on the summation of all past attention probabilities. Hence the coverage vector contains the information of alignment history. The authors append the coverage vector in the attention model so that the decoder is capable to know which part of input image has been attended or not. Let n' denote the attention dimension and q denote the number of output channels of convolution function Q; then $\nu_{att} \in R^{n'}$, $U_s \in R^{n' \times n}$, $U_a \in R^{n' \times C}$ and $U_f \in R^{n' \times q}$. The high-resolution context vector cB_t is computed based on another coverage based attention model f_{catt} with different initialized parameters except the Us transition matrix.

They also illustrate the performance of multi-scale attention model in Fig. 3. The left part of Fig. 3 denotes the visualization of single-scale attention on low-resolution annotations and the right part denotes the visualization of multi-scale attention only on high-resolution annotations. Fig. 3 (a) is an example that the decimal point "." is under-parsed by only relying on low-resolution attention model. However, the high-resolution attention in the multi-scale attention model successfully detects the decimal point. Fig. 3 (b) is an example that the math symbols "- 1" are mis-parsed as "7" due to the low-resolution attention model while the high-resolution attention model can correctly recognize them.

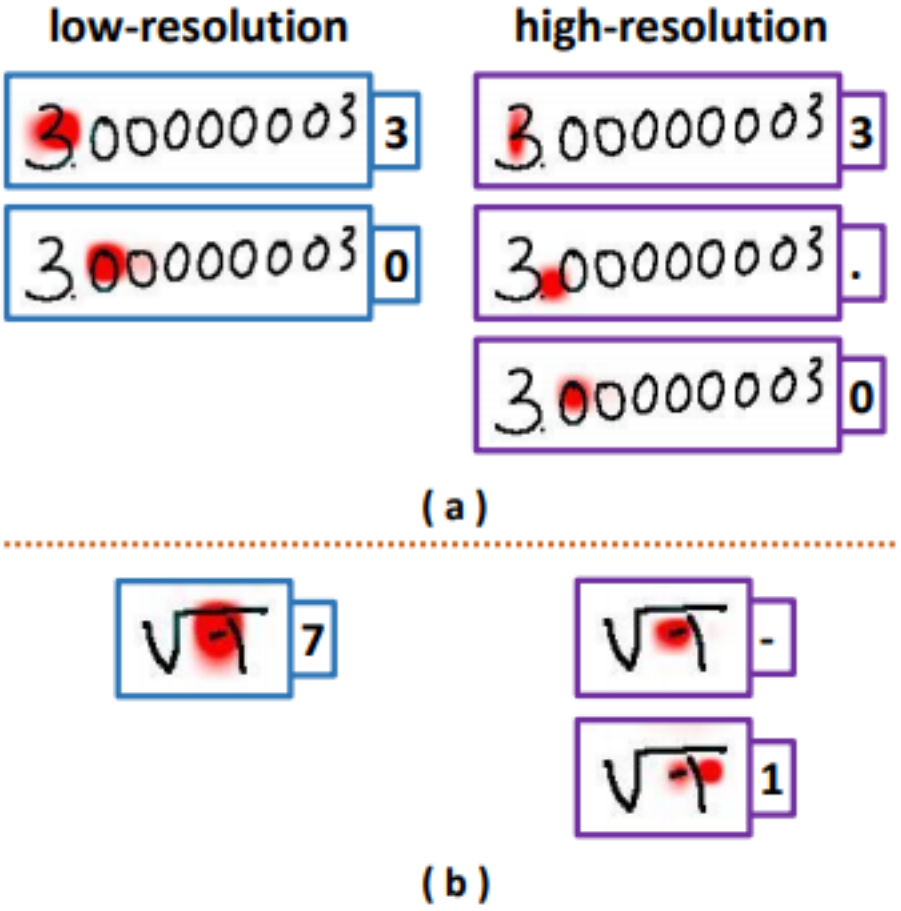

Figure 3: Two examples of attention visualization on low-resolution annotations and on high-resolution annotations. The attention probabilities are shown through red color and the predicted symbols are shown on the right of images.

0.6 Evaluation

1. Evaluation of dense encoder
   The author start the proposed multi-scale attention model with dense encoder from WAP. As shown in Table I, WAP achieves an ExpRate of 44.4% on CROHME 2014 test set and an ExpRate of 42.0% on CROHME 2016 test set.The implementation details of the dense encoder is illustrated by the main branch in Fig. 2. It can be observed that the ExpRate increases about 5.7% on CROHME 2014 and 5.5% on CROHME 2016 by employing dense encoder.

TABLE I
COMPARISON OF RECOGNITION PERFORMANCE (IN %) ON CROHME 2014 AND CROHME 2016 WHEN EMPLOYING DENSE ENCODER AND MULTI-SCALE ATTENTION MODEL.

System	CROHME 2014		CROHME 2016	
	WER	ExpRate	WER	ExpRate
WAP	19.4	44.4	19.7	42.0
Dense	13.9	50.1	15.4	47.5
Dense+MSA	12.9	52.8	13.7	50.1

By the time of preparation of this report, the model had trained till 21 Epochs. The logs are depicted below :

```
Epoch  21 Update  34100 Cost  3.107202751636505 Lr  1.0
Epoch  21 Update  34200 Cost  2.967513539791107 Lr  1.0
Epoch  21 Update  34300 Cost  2.602363543510437 Lr  1.0
Epoch  21 Update  34400 Cost  2.9444775664806366 Lr  1.0
Epoch  21 Update  34500 Cost  3.4290895783901214 Lr  1.0
Epoch  21 Update  34600 Cost  2.953728438615799 Lr  1.0
Epoch  21 Update  34700 Cost  2.99407826423645 Lr  1.0
Epoch  21 Update  34800 Cost  3.1175891482830047 Lr  1.0
Epoch  21 Update  34900 Cost  3.003316488265991 Lr  1.0
Epoch  21 Update  35000 Cost  2.986780835390091 Lr  1.0
Epoch  21 Update  35100 Cost  3.5435642766952515 Lr  1.0
Epoch  21 Update  35200 Cost  2.9221662497520446 Lr  1.0
Epoch  21 Update  35300 Cost  2.9533772969245913 Lr  1.0
Epoch  21 Update  35400 Cost  3.1789787447452547 Lr  1.0
Epoch  21 Update  35500 Cost  2.9613672149181367 Lr  1.0
Epoch  21 Update  35600 Cost  2.95258740067482 Lr  1.0
gen 100 samples
gen 200 samples
gen 300 samples
gen 400 samples
gen 500 samples
gen 600 samples
gen 700 samples
gen 800 samples
gen 900 samples
valid set decode done
Valid WER: 21.36%, ExpRate: 32.69%, Cost: 15.586523
```

0.7 Conclusion

In this study, the authors improve the performance of attention based encoder-decoder for handwritten mathematical expression by introducing the dense encoder and multi-scale attention model. It is the first work that employs densely connected convolutional networks for handwritten mathematical expression recognition and they propose the novel multi-scale attention model to alleviate the problem causing by pooling operation. Also, they demonstrate through attention visualization and experiment results that the novel multi-scale attention model with dense encoder performs better than the state-of-the-art methods.

0.8 Work Distribution

1. Aditi Shrivastava
   - Dense Encoder

2. Bhavi Dhingra
   - Coverage Model
   - Data Preprocessing

3. Hitesh Kaushik
   - Attention Mechanism

4. Ravi Jakhania
   - Decoder