Comparing the Performance of Accessible Deep Learning and Non-Deep Learning Models in Deciphering Scripts

Siddharth Krishnan

Received October 10, 2024
Accepted February 25, 2025
Electronic access March 15, 2025

Deciphering unknown scripts can teach us lots about how our predecessors lived and thought, but how do we find out what they actually said? It usually takes laborious hours of effort to decipher an unknown script, and a suitable method of translation is difficult to come by. What if we could automate the process of reading an unknown inscription? In this paper, we use visual similarity calculations to match characters from an unknown script to characters from a known script (since the study of Daggumati and Revesz suggests that, if there is a clear correlation between a pair of characters from a script, then it is likely that they encode the same information. However, this only really applies to related languages. We test this on the Indonesian languages Javanese (acting as the unknown language) and Balinese (acting as the known close relative). We also test it again using the archaic South Indian script Pallava to try a more distant known relative. Pallava is the ancestor of Javanese and was used 1500 years ago. We try two different models to see which one is more effective: A deep learning model trained on real-world images or a non-deep learning Structural Similarity Matrix. We expected the OpenCV model to do better because of its contrast-focused approach, but the difference between both models was not statistically significant. Additionally, both models were statistically better when comparing Javanese to Balinese (its closer relative) than comparing it to Pallava (its more distant relative), with the difference being statistically significant. This validated our assumption that the correlation between visual similarity and shared meaning weakens as relationships become more distant.

Introduction

Background Context

There are many undeciphered writing systems in the world, like Linear B and the Indus valley script. There have also been several well-known decipherments of scripts that we now know how to read, such as Egyptian Hieroglyphics, which opened the door to Ancient Egypt and helped us understand it better. In that case, the Rosetta stone was instrumental in the effort of decipherment because it provided the same text in both the unknown script and a known script. However, it isn't always easy to find the exact series of characters in an already known language. This creates a big question: Can a simple, accessible automated model read unknown characters just based on their visual similarity to characters in a known script, and how accurate would this be? There would be many benefits to using these simple models which focus on the visual similarity of the characters for decipherment because it would make deciphering scripts much more accessible. Other researchers have also used similar models which compare characters visually to make judgements about the relationships between ancient scripts (Daggumati and Revesz), but they have not been able to phonetically decipher the scripts.

Literature Review

A study by Daggumati and Revesz [1] tries to see how related different languages are by visually analyzing its characters using a machine learning model. The study finds that, when two scripts are of relatively common origin, the characters which look like the closest counterparts usually have the same meaning, but this is not the case when the scripts do not share a common origin. The further apart the scripts are, the less this fact will hold true. For the sake of our research, we will assume that the most visually similar characters have the closest meanings. This means that our model can only be used once a closely related script has been found, which is a serious limitation, but we propose a method of doing so in the section concerning extensions. They used deep learning models similar to TensorFlow, but trained them themselves. We will be using a pre-trained TensorFlow deep learning model, in addition to a non-deep learning model such as OpenCV.

Additionally, a study by Hauer points to a method of deciphering unknown manuscripts by looking at the patterns of different characters in each word and matching these patterns with the model's database of character patterns in known languages. This differs from our method of looking at each character individually. However, it is useful when looking at large manuscripts such as the Voynich manuscript which was used as the example

in the paper[2].

Perhaps Daggumati and Revesz's method can be extended to decipher the unknown script, Javanese, into known characters, so it can be read. Or, studies like Hauer's may lead us to a conclusion that more advanced and technical methods, holistically analyzing large inscriptions to look for large-scale patterns, rather than comparing each character in isolation from the others[1,2].

Aims for This Study

In this case study, we have used the Indonesian script Javanese as the input "unknown" script and the Indonesian script Balinese as the known script. We will run it a second time with the Indonesian script Pallava as the known script. We used one script which is closely related and visually similar (see figures 1 and 2) because we expected the model to be disastrous and wanted at least some correct characters to create the most meaningful analysis possible with a variety of talking points. However, in a real-world context, it would be possible to compare the unknown inscription's characters to any chosen known language and use that chosen known language as a biscript (since an extension to our model would figure out which of the known scripts is the closest to the unknown script using the method described by Daggumati and Revesz[1]). We use two models, VGG16 and OpenCV structural similarity index to aim to accomplish this task since these models can check the similarity between multiple images. The reason for using both these models is because they are both easily available to the public, and this would help represent what those with little coding experience could do. Considering this, we hope to create a model which can assign the correct Balinese equivalent to 3 out of 5 javanese import characters. The balinese data will include 10 characters for the model to choose from, 5 of which are the equivalents of the Javanese characters. However, knowing that these models can only compare whole images, we have chosen to limit our scope to characters inputted individually rather than as a set.

As said above, our models would only work on scripts that have some degree of common origin, so we will take Javanese as the unknown language and compare it with Balinese and Pallava. Pallava, as seen on the figure, is the direct ancestor of Javanese, but it will still not be as closely related to Javanese as Balinese (the closest relative overall). Due to this, we still expect the model to struggle with Pallava in relation to balinese.

Method

Summary

In simple terms, each model is running a series of image similarity checks. There is the input of the unknown characters, listed as individual PNG images, and the list of all the characters

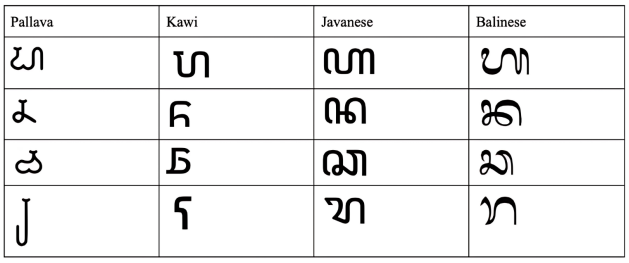

Fig. 1 Here is a figure of the evolution of the scripts. Both can be traced as far back as the South Indian Pallava script. That was then modified and derived into Kawi, which then led to the creation of both Javanese and Balinese. This explains the close relationship between Javanese and Balinese because they only diverged from Kawi in the 16th century[3,4].

belonging to the chosen known "matrix" language, which is the language used for decipherment. In the case of this study, the input characters come from the Javanese language and the chosen matrix language is Balinese. Therefore, each of the inputted Javanese characters will be compared to all of the Balinese characters, and each Javanese character will get a corresponding Balinese Character which the model thinks is the most visually similar to the input character. Additionally, if desired, it is possible to view the entire set of data, as shown later in the results section. The subsequent 3 figures are enlarged to make the characters easier to view. These images also serve as a visual representation of the ground truth (when characters are matched correctly).

Choice of Data

Fig. 2 The selected Javanese characters used for input (as separate PNG files). They read 'a','hi','ca','krA', 'n', 'o', 'mu', 'ce', 'prA', 'k' from right to left[3]

Fig. 3 the Balinese equivalents of the Javanese inputs[3]

Fig. 4 the Pallava equivalents of the Javanese inputs[3]

The rationale behind choosing these specific characters (since they mean nothing in either language) is due to the characters themselves. There are five major categories of characters in both scripts: Vowels on their own, consonants with vowel marks, consonants with the default 'a' vowel, consonant clusters with vowels, and consonants with no vowel at the end. It is important to note that the representatives from the consonant-and-vowel category include short vowels, and the representatives from the consonant-cluster-and-vowel category include long vowels. This was done so both short and long vowels were represented. Areas (similarity of the image components themselves) will both increase mean structural similarity.

The most significant thing to note is that, in the Javanese input set, each category is represented once, ('a' is the standalone vowel, 'hi' is the consonant-plus-vowel, 'ca' is the consonant with the default 'a' vowel, 'krA' is the consonant cluster with a long vowel, and 'n' is the consonant without a vowel. However, in the Balinese/Pallava database set, each category is represented twice. This is so that various types of characters are considered. The Javanese input set contains all the equivalents of the Balinese set described, but only five Javanese characters at a time will be used as an input. We will test 10 random combinations of five input characters (chosen out of the 10 possible inputs stored in the set). Both of the models will get the same set. The accuracies from each of these sets will be averaged. Both models will get the same input sets. This process will be done by comparing Javanese with Balinese first, then Pallava.

Technical Walkthrough of Both Models

Now, it is important to discuss the way that both of these models work. The first model is VGG16 used in the VGG16 model and ImageNet weights. The '16' in VGG16 stands for the fact that the convolutional neural network used by VGG16 has 16 layers. Each one uses filters and weights to help understand and classify the images. The model uses ImageNet weights. ImageNet is a collection of 14,000 images, and the imageNet weights are used by models trained on this large set of images[5,6]. This practical experience makes this a good fit for our scenario.

The second model used is the model made by OpenCV and calculates mean structural similarity, proposed by Wang et al in 2004. In this system, structures in the images (which in our case are shifted to grayscale) are defined as parts of the image where there is large contrast with the surrounding areas. The structurality of an image increases as there is more contrast (in the same places as the original) and decreases as there is less. In other words, two images with a similar quality (contrast between structures and the background) and a similar positioning of structuring will have a higher score[7]. It is important to remember that, in our case, we are dealing with characters written in black on a white background so contrast would easily be discernible by the model.

Both models rank similarity on a scale from -1 to 1. The number of instances where the most similar character matches with the ground truth will be termed as the model's accuracy and function as the dependent variable in this paper. The independent variable is the model that we will be using (OpenCV vs VGG16).

Our hypothesis is that the OpenCV model will do better than the VGG16 model. This is because the OpenCV model works by using areas of high contrast (like black marks on a white surface) to designate the structures used for similarity comparison. On the other hand, VGG16 was trained on ImageNet, a set of real-world images. These real-world images would be multi-color and complex, compared to script characters which are always black and white, and are much simpler than real world images. For this reason, we expect the OpenCV model to do better overall because its method of looking at contrast (black on white) would be more suited to comparing characters from scripts. Additionally, we expect the accuracy for both models to be higher when comparing Javanese with Balinese than when comparing Javanese and Pallava, because Pallava is not as closely related to Javanese (see above)

Results

These are the results of our study. It is important to remember that there are ten total Javanese characters in the input set, and the corresponding set of characters in Balinese and Pallava, the two scripts we will compare with the input script. However, in each of ten trials, a random set of only five inputs will be used at a time.

OpenCV - Balinese	Tensor flow - Balinese
3	4
2	2
2	3
2	3
2	2
2	2
1	2
2	2
1	2
1	2

Fig. 5 the data from the two models comparing Javanese and Balinese. The ten columns represent each of the ten trials, where a random five of the Javanese characters were compared with the Balinese set. The numbers represent how many characters (out of the 5 total inputs), each model guessed correctly.

When looking at the results, there are two important questions. First, which model makes more accurate comparisons between Javanese and Balinese, and Javanese and Pallava? Is this statistically significant? Second, for each model, does Ba-

OpenCV - Pallava	Tensor flow - Pallava
0	4
0	2
0	3
1	3
0	2
0	2
1	2
0	2
1	2
0	2

Fig. 6 The same data from Both models, but comparing the Javanese inputs with Pallava characters instead of Balinese.

linese or Pallava make more accurate comparisons when used as a "known language" for Javanese? The claims made in the hypothesis were that OpenCV would make more accurate comparisons than TensorFlow regardless of the known language, and that comparing Javanese with Balinese would lead to more accurate comparisons than comparing it with Pallava. We can analyse both the real-world style data with ten trials of 5 inputs at a time, and the more analytical data of each possible input character compared with each possible output character. Let us begin by answering the first question.

At first glance of the data of figures 5 and 6 (which uses multiple trials of five randomly selected inputs), TensorFlow clearly outperforms OpenCV by having a higher mean of accurate comparisons with both Balinese and Pallava. Looking at the standard deviations and the 5-point summaries of the datasets also supports this conclusion. These observations go against our hypothesis, but are they statistically significant? We have decided to perform a two-tailed Mann-Whitney U Test to find out. If we conduct the test on the two columns of figure 5, the accuracy of Javanese-Balinese comparisons, we find the p-value to be 0.123, which is greater than the level of significance 0.05, This means that the null hypothesis, which would state that the difference between the OpenCV Balinese data and the TensorFlow Balinese data is not statistically significant, is retained. When comparing the OpenCV Pallava and TensorFlow Pallava data (two columns of figure 6), the p-value of the test is 0.393, which is also greater than the level of significance, 0.05, also retaining the null hypothesis[8].

This refutes our first hypothesis, since the U Test proves that the difference between the two models' performance is not statistically significant, and any minute difference is in the favor of Tensorflow when looking at the surface-level statistics. Now, let us investigate the second question:

For this, we need to compare the Balinese and Pallava result sets for both the models. This means comparing each column on figure 5 with its corresponding column on figure. Looking at the data, as well as surface statistics like the mean, another clear conclusion can be made. This time, it favors our hypothesis: The Javanese-Balinese comparisons are indeed much more accurate than the Javanese-Pallava comparisons. Again, we use the Two-Tailed U Test to check if the difference is statistically significant. When comparing OpenCV's Pallava with its Balinese data, we find that the U Test's p-value to be less than 0.001. This is less than the level of significance, 0.05. This means that the null hypothesis, which states that the difference between Javanese-Balinese and Javanese-Pallava comparisons is statistically significant. If we use the TensorFlow Balinese and TensorFlow Pallava data, we get the same p-value, meaning the null hypothesis is also rejected. This means that our hypothesis that Javanese-Balinese comparisons will be more accurate than Javanese-Pallava comparisons is true.

Conclusion

In summary, the difference between the accuracy of TensorFlow's comparisons and OpenCV's comparisons is not statistically significant. Additionally, both the models' success rates stayed under 50% for both languages. However, it is true that comparing Javanese and Balinese will lead to more accurate comparisons than comparing Javanese and Pallava. This is important because it shows that the principle of Daggumati and Revesz[1] also applies to actually deciphering unknown scripts and making them readable.: The closer the relationships between the characters, the more accurate it will be to use visual similarity to decipher them.

Limitations

Regarding the limitations to our specific methods, one major factor comes to mind. Our method has used screenshots of each of the characters to create the PNG images. This is due to lack of availability of pre-prepared image sets of characters belonging to lower-resource languages like the ones used in this case study. However, the screenshotting process can create many opportunities for error, especially when the entirety of the character was not in the frame. In the case of the VGG16 model, the part of the character touching the edge was described as a feature in the model, which it could not find in the other characters which were entirely within the frame. This threw off the model, leading it to make its guess based on the presence of this feature. This is one among many of the inconsistencies that screenshotting would provide. Lack of proper cropping of the screenshots could also lead to a distortion of the image since too much white space may be recorded by the models as a feature, leading again to a distorted guess.

Additionally, the model can only compare related scripts due to its assumption that close visual similarity correlates to shared meaning. However, the rationale for this is that, since these models are not purpose built (which also explains their low

accuracy and impractical usage), we thought that we should go a bit "easy" on them. Having purpose-built models (or linking these models to ones that do other stages of the decipherment process, like finding related scripts), would have allowed us to test more real-world scenarios.

We also only test a very small set of characters. This means that the characters get recycled in the input data (since there are ten total input characters and only two mutually exclusive sets of five characters). This also slightly reduces some of the statistical validity of our data.

Extensions

One of the possible extensions of the method is to connect either of these models with another model (as mentioned above) that factors in linguistic meaning in the texts. For this, it would be assumed that inscriptions (with the exception of proper nouns) mean something in the language they are written in. Therefore, it is possible to cross-check all the outputs against a dictionary of the known language. For example, even though the testing input set we used is totally meaningless (being specifically designed to test the computer), 'ahicakrAn' would probably be close to Balinese words like 'ahimsa', and all the given outputs could be whittled down based on how much they resemble real Balinese words.

The second possible extension to the method, other than simply adding more characters to both the input sets and the known language sets, is to allow the model to take in PNG images containing multiple characters and split them based on where it thinks one-character ends and another begins. However, the difficulty with this is that characters often contain multiple detached strokes, so we would have to train the model to group those together to compose a whole character. For basic, off-the-shelf models like the ones used in this study, that may be quite a challenge. However, it would make the model even more usable because characters do not have to be separated before entering the model.

We could also make the VGG16 model more effective by actually training it on the known scripts instead of using just the generic ImageNet weights to train the model, increasing the skill of the model when it comes to discerning similarities between known and unknown scripts. It would also help us with more advanced work such as dealing with noisy and distorted images, since one limitation was the simplicity of the dataset used.

Implications

Neither of the models has a high enough success rate to be useful in the field. This shows that generic, off-the-shelf architectures like these should not be trusted for the purpose of deciphering scripts. Instead, models trained for the express purpose of script decipherment should only be used. This also shows that technology must be rigorously tested before being used for script decipherment. Additionally, using the wrong technology to decipher a script may give researchers a false sense of security (if the ground truth is unknown), even though the decipherment itself would be very incorrect. However, it is still important to note that while visual similarity may be too basic to decipher scripts, it can help uncover theoretical principles of decipherment, such as the one mentioned in the conclusion of this paper.

References

1 S. Daggumati and P. Revesz, *Convolutional Neural Networks Analysis Reveals Three Possible Sources of Bronze Age Writings between Greece and India*, 2023, Online. Internet. 22 Sep. 2024. Available: www.researchgate.net/publication/369916621_Convolutional_Neural_Networks_Analysis_Reveals_Three_Poss

2 B. Hauer, *Computational Decipherment of Unknown Scripts*, 2016, https://era.library.ualberta.ca/items/1ff57e8b-825a-40bb-b1b6-941c76f97a3f.

3 V. Rajan, *Aksharamukha*, 2018, https://www.aksharamukha.com/converter.

4 I. D. A. Bagus, *Contemporary Use of The Balinese Script*, 2003, Online. Internet. 22 Sep. 2024. Available: www.unicode.org/L2/L2003/03118-balinese.pdf.

5 Y. LeCun, Y. Bengio and G. Hinton, *Deep learning*, 2015.

6 R. G., *Everything You Need to Know about VGG16*, 2021, Online. Internet. 29 Sep. 24. Available: medium.com/@mygreatlearning/everything-you-need-to-know-about-vgg16-7315defb5918.

7 Z. Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, *Image Quality Assessment: From Error Visibility to Structural Similarity*, 2004.

8 *Statisty*, https://statisty.app.