Developer Stories · TensorFlow Lite

KingSoft WPS: document image dewarping based on TensorFlow

December 10, 2019

A guest post by Longfei Xiong, Cheng Du, Ronghua Chen, Hui Zheng and Xuhua Hu from WPS AI Engineering Team

Machine learning use cases at KingSoft

WPS Office from Kingsoft is a productivity tool that serves 150M+ users globally. We strive to provide the best tools and features to our users so that they can get their document processing done efficiently. Using TensorFlow, we can help our users to process their documents in a more intelligent manner in a number of use cases, including:

- **Document dewarping:** automatically flatten a warped document image. We will elaborate on this in the next section.
- **Document edge detection:** the CNN-based rectangle detection network devised at WPS can quickly identify the edges of target document, solely on-device.
- **Automatic document category classification:** we use TensorFlow Lite to implement a machine learning model that is able to automatically classify documents based on document images, and provide corresponding functionalities based on the category.

localize texts quickly and accurately for users who are sensitive to data security.

- **Layout analysis:** when converting images to documents, we use both TensorFlow and scikit-learn to implement layout analysis; these frameworks considerably reduced our engineering efforts.

Fig 1. Document layout analysis by WPS

Note that all these functionalities happen behind the scenes so we encourage readers to download our app on Google Play and Apple App Store, and give them a try.

Today, we will focus on just one use case - document image dewarping.

What is document image dewarping?

When reading a book, many of our users run into issues when they want to then extract an excerpt for personal notes. Because the book is fairly large in size, the particular page cannot be fully flattened, which makes it a challenge to take a quality picture for reading later.

warped book page

Fig 2. A warped book page
(Note that the texts are not comfortable to read, given the curvature of the page)

Presumably, you cannot always carry a scanner with you. So, what could we do in this situation? Some tools on the market can perform perspective correction on an image from a distorted perspective like below:

Fig 3. Perspective correction example

This helps users in certain cases, but it only solves the problem of perspective correction. If there is curvature in the image, the correcting effect is still not satisfactory.

We paid considerable attention to this user pain point and wanted to help our users by image with a curved page, and show a perfectly flattened image to the user as seen below (we even automatically crop the image for the user and remove the background).

Fig 4. Dewarping effect

The processed image now looks significantly nicer and is much easier to read and consume. Having the ability to take notes or share this kind of image with friends/colleagues makes our users more productive and effective.

Moreover, curvature correction is very helpful for downstream tasks such as OCR because distorted characters often confuse OCR algorithms and cause recognition failures. By dewarping the image, we are able to dramatically increase the accuracy of our OCR feature.

Fig 5. How dewarping improves OCR

Now that you have a better sense of what document image dewarping does and why it's useful, let's discuss how we made it happen with the help of TensorFlow.

On-device ML + server-side ML for document dewarping

Currently mainstream approaches of document dewarping, and our method as well, all follow the same pattern:

- First, detect the edges or the four corners of the document.
- Second, conduct geometric correction by performing perspective changes.

On-device edge detection

would be much easier to implement. However, running on-device provides a better user experience since the user can see the detection result in real time. So, we pushed the envelope and implemented this on-device. You can see our real time detection demo below:

Fig 6. Edge detection demo

At a high level, our implementation of the edge detection step works as below:

1. We use tf.keras to build a deep neural network to detect document edges. In our image scanning algorithm, the size of our CNN document detection network is roughly 5M. The backbone network we use is SqueezeNet (TensorFlow implementation is open sourced here).
2. We use fit_generator from the tf.keras module and the tf.data module to preprocess the training data. We also customize the cost function according to our business requirements.
3. After training, we use the TFLiteConverter that comes with TensorFlow Lite (version 1.x) to convert our model into FlatBuffer format and integrate it into our apps.

When the user turns on the camera for preview, our deep learning model runs in the background, detecting the document area frame-by-frame and rendering the detection results on the screen. To achieve this in real time, we had to deviate from traditional edge detection algorithms and resort to deep learning. As we iterated on our deep learning model, our model's performance kept improving, but the size of our model was also getting larger, which demands more computational power. In order to accommodate mid and low-end mobile phones, we simplified our model by emulating the structure of SqueezeNet and designing a convolutional neural network with only 7 layers instead of the much deeper network we initially used. Although the new model is shallower, there is only a slight loss of accuracy, and the speed and memory usage improved significantly. This trade-off allows us to run the detection model on low-end phones at the speed of several frames per second while on high-end devices at 30 frames per second. Finally, we add a filter outside the neural network to compensate for the jitters caused by inaccurate detections, which makes the detection results visually more stable.

After completing the detection model and coding, we found that the size of our internal SDK package is a bit too large. In addition to optimizing our own code and dependencies, we needed to streamline the inference framework. We needed a fast, lightweight mobile inference framework. We conducted a lot of evaluations and benchmarks, and eventually settled on TensorFlow Lite. We choose TensorFlow Lite because it has the following advantages:

- **Small size**: our trained ML models can easily be shrunk in size and delivered to mobile devices. The memory footprint is also small, so it is very friendly to our users.
- **Cross-platform support**: TensorFlow Lite inherently supports both iOS and Android, and we have verified its stability on dozens of mobile devices.
- **Easy model conversion**: training models using TensorFlow and converting them to TensorFlow Lite models was very smooth for us.
- **Natural adoption**: TensorFlow Lite's architecture design and workflow is very similar to TensorFlow inference, and it is small and easy to use at the same time. So it was natural for us to adopt it.
- **Vibrant community**: TensorFlow has a very active community and rich community resources, so we are able to learn from our machine learning peers and find valuable references for our use cases.

Overall, adopting TensorFlow Lite enabled us to focus more on on-device algorithm development and model optimization.

Geometric correction on the cloud

Geometric correction in the second step requires massive computational power, and it is not feasible to conduct it solely on-device at the moment. Therefore, we deployed it on our high performance clusters. We compress and encrypt the original images, transmit them to our servers, dewarp the images, and send them back to users' mobile devices to display. In the future, we will explore how to move this part from the cloud to on-device as well.

Fig 8. WPS document dewarping feature demo

Conclusion

In this post, we discussed how we take a hybrid approach (on-device and cloud ML) to develop the documentation image dewarping feature inside the WPS app (Android and iOS), based on the ML model complexity, hardware capabilities, and business requirements. We believe that as hardware becomes more powerful, more and more ML workloads will run solely on mobile devices. On the other hand, the arrival of 5G technology will remarkably increase bandwidth and reduce network latency, and hopefully make cloud ML more accessible and user-friendly. We look forward to them and will keep improving WPS products based on the latest technology advancement.

Developer Stories TensorFlow Lite

Next post

KingSoft WPS: document image dewarping based on TensorFlow

French

Introducing TensorFlow Videos for

December 06, 2019 — When the TensorFlow YouTube channel launched in 2018, we had a vision to inform and inspire developers around the world about what was possible…