Smart TeamBoard: a Knowledge Sharing Progressive Web Application Supported by Efficient Convolutional Neural Networks

Sandor Kiraly[a], **Daniel Csutoras**[b]

[a]Eszterházy Károly University
kiraly.sandor@uni-eszterhazy.hu
[b]Eszterházy Károly University
danielcsutoras@gmail.com

Abstract

Progressive web applications (PWAs) are websites that look and feel like an app which means that users can access all information and capabilities without downloading a mobile app.

Instead, progressive web applications use modern web technology to deliver app-like experiences to users, right in their browsers. Our PWA allows registered users to share their resources (photos, videos, sound files, URLs, etc.) by theme, adding them to different pinboards that are located on the user's board. Users can also search for shared images that are tagged by a pre-trained EfficienNet that is the default Convolutional Neural Network of the application.

The application allows users to train new CNN models by uploading tagged images and using transfer learning. Our PWA can therefore also recognise images that belong not to an ImageNet but a new, user-created class. The training process is parameterized: users can choose a base model and can change the number of layers, activation functions, training methods, loss functions, etc. These neural networks pretrained by users are organised in different tree structures, for example, a sport car classifier model gets into a tree structure that is rooted in a car recognition model. During the recognition, initially, the base EfficientNet CNN gets the uploaded image, followed by the root models of the other CNN tree structures. If a root model recognizes the image, the additional CNNs in the tree will receive the image.

Our PWA provides optimizing CNN models for deployment and execution by reducing the number of parameters and operations involved in the computation by removing connections, and thus parameters, in between neural network layers.

Keywords: progressive web application, convolutional neural network, transfer learning, model scaling

MSC: 68T45, 68U35

1. Introduction

In recent years, there has been a paradigm shift from browser to native apps and then back to browser again after Google announced a new technology, named Progressive Web Application that can take advantages of new features supported by modern browsers, including service workers and web app manifests, that let users upgrade web applications to progressive web applications in their native operating system. Now, even multinational companies like Twitter, Nikkei, Trivago or Facebook have developed PWAs ([5] [23]). What exactly makes a Web app a Progressive Web Application is not clearly defined.

1.1. Detecting the features and working of Progressive Web Applications

PWAs are mobile and desktop web applications that are accessible in any web browser and provide additional capabilities including offline support and push notification. They use Service Worker API, they can be installed to a home screen ([18]), and should load immediately, regardless of network state. According to a Google checklist[1] a PWA makes sites available over HTTPS, the served pages are responsive on tablets and mobile devices, all app URLs load while offline, metadata is provided for Add to Home screen and each page has a URL.

In consequence, a web application can be considered PWA if it requires one or more of the Service Worker APIs, is a server over HTTPs, and it comes with a web app manifest for declaring app metadata like its name, icons and base URL.

A PWA has the following features: (i) they provide an installable, app-like experience on desktop and mobile that are built and delivered directly through the web. (ii) PWAs work for every user, regardless of browser choice because it is built with progressive enhancement as a core tenet, (iii) in addition, they still have ability to load and work at least to some extent, even when the device is offine ([19]). (iv) They are responsive since they fit any form factor: desktop, mobile, tablet. (v) PWAs are also connectivity-independent since they are enhanced with service workers to work offline or on low-quality networks. (vi)Besides, they make services available through HTTPs to prevent snooping and to ensure content has not been interfered with and make re-engagement easy through features like push notifications. (vii) In addition, PWAs can share the application through URL, and do not require complex installation. (viii) They have capability to estimate the available storage that an application already uses and to know the available quota enforced by the browser ([23]). (ix) PWAs guarantee data is stored continuously to avoid the browser being purged without user contribution even if the memory is running out ([23]).

At the core of this new technology is the concept of service worker which is a set of APIs that allows developers to cache programmatically and preload assets and data, manage push notifications and others. Technically, a service worker is a JavaScript

[1] https://developers.google.com/web/progressive-web-apps/checklist

module running in its own thread and providing generic entry points for event-driven background processing, for example, reaction to the receiving of a push notification.

The Service Worker is a special type of web worker which is implemented in a dedicated JavaScript file and runs in a separate thread with respect to the main JavaScript thread. Intuitively, a service worker can be considered to be a piece of JavaScript code running in parallel to the main page, providing persistent background processing, and interacting with the rest of the PWA in an event-driven fashion. A service worker can listen to events dispatched from the main page (for example, the fetch event is raised when the main page makes network requests) or other sources (for example, the push event is raised when a push notification is received from a remote server). When the service worker is installed a set of static resources is added to a local cache and when the main page makes any network request (e.g., an Ajax call), then the service worker intercepts it, leading to two cases: a) the requested resource is related to "an important program", so it is always directly performed; b) otherwise, a cache of previous requests is queried and the cached object is responded to, otherwise the network request is actually performed, its result is put into the cache for future use and returned as a response to the main page.
The uploaded pictures in our Smart TeamBoard are tagged by different CNNs that makes images searchable.

1.2. Convolutional Neural Networks in Computer Vision

CNNs (ConvNets) were first successfully applied to relatively small tasks such as digit recognition ([24]), image interpretation ([2]) and object recognition ([15][4]). In this time, their size was greatly limited by the low computational power of available hardware. Since 2010, however, CNNs have greatly profited from Graphics Processing Units (GPU). In 2011, the first implementation [3]) of GPU-based ConvNet on the CUDA parallel computing platform has been introduced [17]). Recently, CNNs are important deep learning models that have achieved great success in image classifications ([13][28] [22]), speech recognitions ([20][1]) and natural language understanding ([30][29] [16]).

While ConvNet models have become more accurate by becoming bigger, the training cost of CNNs is extremely high for two reasons. Firstly, CNNs are getting more complicated due to increased depth and parameters. Since AlexNet ([12]) won the 2012 ImageNet competition, and it has only 8 layers and more than 60 million parameters, the 2017 ImageNet winner SENet([8]) achieved 82.7% top-1 accuracy but with 145M parameters. Training these large-scale CNNs requires thousands of iterations of forward and backward propagations, and therefore is much time-consuming.

Secondly, the training samples are getting much larger. One of the early CNNs, LeNet-5, was trained to recognize handwritten digits on a MNIST data set, which contains only 60,000 images in the training set and 10,000 images in the testing set ([14]). In contrast, a larger dataset called ImageNet was provided in 2009, including more than 1.2 million high-resolution images.It seems we have already reached the hardware memory limit, and thus further accuracy gain needs better efficiency.

We can say that deep Convnets are often overparameterized. It is not surprising therefore, that several studies have achieved good results in Model compression

([7],[8][27]). This is a very common way to reduce model size by trading accuracy for efficiency. Efficient mobile-size CNNs have been made after reducing the parameters of base CNNs such as SqueezeNets ([11],[6]), MobileNets ([9],[21]), and ShuffleNets ([31]).

2. Implementation details of Smart TeamBoard

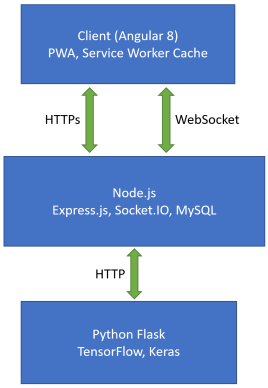

Figure 1: Operational diagram of Smart TeamBoard

Smart TeamBoard (ST) is a client-side Angular 8 application and a small, simple Node.js server (see Figure 1). The server handles user authentication, constructs the initial state of the app. Once loaded in the browser, the app requests data directly from the ST API. The simplicity of this basic architecture has helped us to deliver exceptional service reliability and efficiency at scale. Data apart from CNN models are stored in a MYSQL database using thirteen tables.

The server was built with Express.js which is the most popular, minimal and flexible Node web framework. The application also uses Socket.IO which is a library that enables real-time, bidirectional and event-based communication between the browser and the server. If the client is compatible with Socket.IO, it uses WebSocket protocol under the hood. To handle high loads the server uses multiple Node processes with the Node Cluster module. Socket.IO instances are synchronized with Redis[2].

To maximize efficiency the Angular client uses a custom preloading strategy which insures that if the client has a slow network connection the application will use lazy loading, otherwise all modules will be preloaded to achieve high performance. The client applies several modern features like efficient typeaheads, infinite scrolling on long lists and animations to provide smooth user experience.

To avoid the overloading of the Node.js server, another server has been added to the application with the task of training and storing models and the recognition of uploaded images by using the Flask framework. This server communicates with the

[2]Redis is an open source, in-memory data structure store, used as a database, cache and message broker.

Node.js server through HTTP. For the same reason, the server training tasks are handled in a task queue implemented with RQ and Redis.

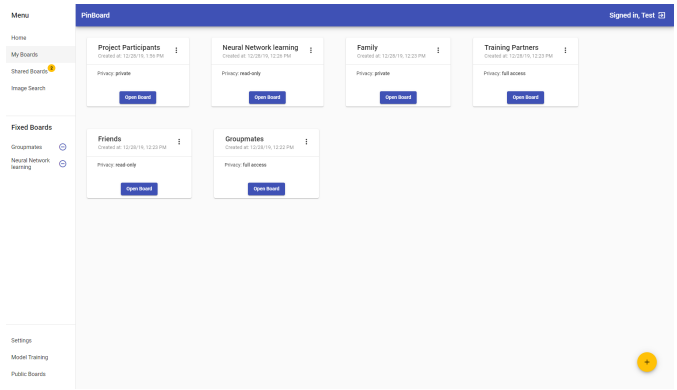

Figure 2: User's smart boards

All registered users can create private and shared boards and add their resources (documents, URLs, videos, sound files, pictures, etc.) to them using the drag and drop technique. It is possible to sort resources by dragging them around the board. Users receive a Push Notification if another user shares a new board with them. Our application provides a preview for URLs and tags for the uploaded images. If a user uploads an image the system tries to recognise it. At first, the base, trained by ImageNet pictures EfficentNet-B0 gets this image at which point the others root CNNs. If any of them recognises the picture it will be tagged by this model and the other CNNs in the tree attempt the recognition. In the case of success, another tag will be assigned to this image. There is an option called Advanced Prediction where users can choose in which tree they would like to run the recognition. It is also possible to choose Combined Prediction when the recognition runs in all trees and tries to attach as many tags as possible to the image. The recognistion process runs in the background therefore the tagging is performed asynchronously.

Our application contains only one pre-trained CNN at lunch but users are allowed to train new models. Users can upload their training data in a zip file. The file must contain the images organized in folders by classes and the associated class names for each output. Users also need to select the tree and the node which they want to add the new model to, which will be able to recognize objects in images using transfer learning. Figure 3 shows how a disjoint graph contains CNN models organised in trees. For example, if a user wants to create a male/female recognition model, the new CNN will be inserted into the Humans tree.

It is also possible to close a branch of a tree if, for example, it is no longer worth inserting a new model under the Ferrari Types model. The teaching process can be customized. The user can choose a base model (e.g. EfficientNet-B0, EfficientNet-B4) and set training properties like the number of epochs and toggle Image Augmentation. The user can dynamically add layers on top of the base model and specify the number of

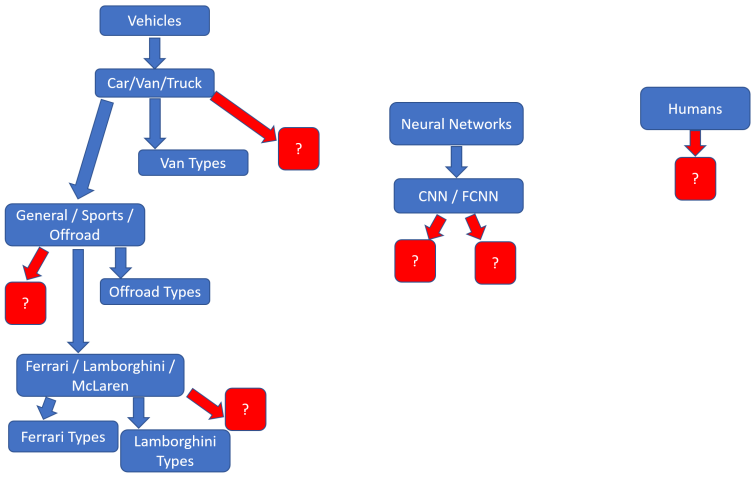

Figure 3: CNNs organised in graph structure

units, dropout rate and activation functions. If it is enabled, augmentation is performed before the training process begins. If the training accuracy is over 85%, the system will save the model. Users have the possibility to delete their models if they are leaf nodes.

Users can search among their uploaded and the shared images. All the tags belonging to an image will be displayed. Figure 4 shows the result of "offroad" search.

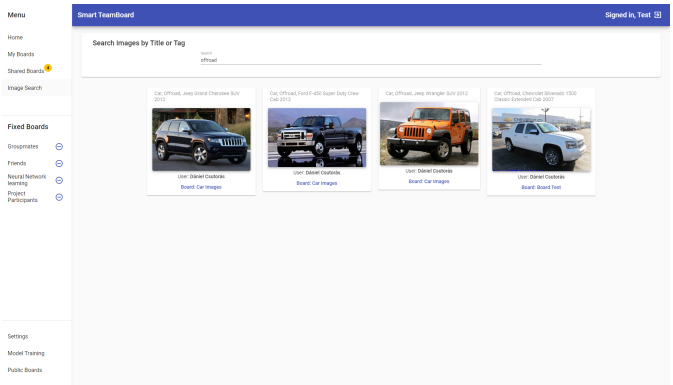

Figure 4: Result of "offroad" search

3. The applied ConvNet in Smart TeamBoard

We applied ResNet50 and Inception V3 models for the first time in the application, as they performed best on "neutral" images. Users could choose from two models during

the training. Because the tagging of uploaded images can be a very slow process in the case of many ConvNet trees in the graph, we reduced the size of our models by applying model compression to avoid a significant reduction in accuracy.

We performed magnitude-based weight pruning with Keras to eliminate the unnecessary weight values in the tensors. Table 1 shows the result in the case of V3.

Our PWA applying V3 model recognises an image in 7.9 seconds with 99% accuracy. The training time for such a model was 3 minutes for a given dataset. Although after 50% pruning, the recognition process became faster, accuracy dropped to 93%, while the training time increased to 7 minutes for the same dataset. However, after 90% pruning, recognition did not get faster, but training time was significantly slower.

Applied model type	Prediction time in PWA	Prediction time of the model	Accuracy	Training time	Number of parameters
V3	7.9s	1.08s	99%	3 minutes	48,803,873
Pruned V3 (50%)	6.6s	0.92s	93%	7 minutes	24,428,325
Pruned V3 (90%)	7s	0.91s	94%	27 minutes	4,049,564

Table 1: Model compression results

We also tested our PWA by applying TensorFlow Lite (TFL) which is designed to execute models efficiently on mobile and other embedded devices with limited computing and memory resources by using 8-bit quantization. Our converted TFL V3 model prediction time was 0.25s which is impressive. We have also tested the accuracy of this model for our dataset. Despite the 8-bit quantization, it achieved 73.1% accuracy. Nonetheless, we declined to apply this. While V3 models recognised some images from our dataset with above 80% or 90% accuracy, the converted TFL model did the same with 0,14% or 0,39%.

Tan and Le ([26]) published the EfficientNet models that have achieved state-of-art accuracy on a different dataset (CIFAR-100, Flowers, etc.) while being 8.4x smaller and 6.1x faster on inference than the best existing ConvNet after applying their own scaling method. According to our test, the EfficientNet-B0 prediction time was 0.7 which is faster what our pruned V3 model achieved. In our PWA, it took 5 seconds with 100% accuracy. The training time was only 2.5 minutes. Table 2 represents the performance of the applied ConvNets in our PWA.

4. Conclusion

The described PWA can be used for storing and sharing resources by registered users. They are notified immediately of shares that concern them. The uploaded images are tagged by CNNs organised into tree structures and stored in a database, which makes it possible to search for different images. Application users can teach models to recognise their own images, which will be inserted into the appropriate model tree structure. The

Applied model type	Prediction time of PWA	Prediction time of the model	Accuracy	Training time	Number of parameters
V3	7.9s	1.08s	99%	3 minues	48,803,873
Pruned V3 (50%)	6.6s	0.92s	93%	7 minues	24,428,325
Pruned V3 (90%)	7s	0.91s	94%	27 minues	4,049,564
EfficientNet-B0	5s	0.7s	100%	2.5 minutes	5.3M

Table 2: Model compression results

training process can be parameterized, providing more efficient training results to the users. Our application uses state-of-art ConvNets that work not only with high accuracy but reduced size.

5. Future Work

The CNNs used in our application provide only object recognition. For example, if there are 2 car types and a man in a picture, it can only get multiple tags during the tagging process if each CNN in the root of CNN trees receives the image, so we do not stop the process after the first tag is assigned to the image. Therefore, we would like to implement semantic segmentation where it is possible to classify each pixel as belonging to a particular class which may make the tagging process faster.

It is planned to provide a faster First Contentful Paint indicator and optimize the application for search engines using server side rendering with Angular Universal. This way the application will be pre-rendered on the server and able to return static html content for the browser to display before the client side Angular application is bootstrapped.

Acknowledgements The research was supported by the grant NNKP-19-2 "New National Excellence Program" of the Ministry of Information and Technology.

References

[1] AMODEI, D., ANUBHAI, R., BATTENBERG, E., CASE, C., CASPER, J., CATANARO, B., CHEN, J.,CHRZANOWSKI, M., COATES, A., DIAMOS, G., ELSEN, E. Deep Speech 2:End-to-EndSpeech Recognition in English and Mandarin, *arXiv:1512.02595, 2015.*

[2] BEHNKE, S. Hierarchical Neural Networks for Image Interpretation, *vol. 2766 of Lecture Notes in Computer Science, Springer, 2003*

[3] CIRESAN, D.C., MEIER, U.,GAMBARDELLA, L.M., SCHMIDHUBER, J. Deep, big, simple neural nets for handwritten digit recognition, *Neural Computation, vol. 22, no. 12, pp. 3207–3220, 2010.*

[4] CIRESAN, D.C., MEIER, U.,GAMBARDELLA, L.M., SCHMIDHUBER, J. Flexible, high performance convolutional neural networks for image classification *International Joint Conference on Artificial Intelligence, 2011, pp. 1237–1242.*

[5] GALLAGHER, N. How we built Twitter Lite *https://blog.twitter.com/engineering/en_us/topics/open-source/2017/how-we-built-twitter-lite.html (2017)*

[6] GHOLAMI, A., KWON, K., WU, B., TAI, Z., YUE, X., JIN, P.,ZHAO, S., AND KEUTZER, K. Squeezenext: Hardware-aware neural network design *ECV Workshop at CVPR'18,2018*

[7] HAN, S.,MAO,H.,AND DALLY, W. J. Deep compression:Compressing deep neural networks with pruning, trainedquantization and huffman coding,*ICLR, 2016*

[8] HE, Y., LIN, J., LIU, Z., WANG, H., LI, L.-J., AND HAN, S.AMC Automl for model compression and accelerationon mobile device. *ECCV, 2018*

[9] HOWARD, A. G., ZHU, M., CHEN, B., KALENICHENKO, D., WANG,W., WEYAND, T., ANDREETTO, M., AND ADAM, H.Mobilenets:Efficient convolutional neural networks for mobile vision applications *arXiv preprint arXiv:1704.04861, 2017.*

[10] HU, J., SHEN, L., AND SUN, G.Mobilenets:Efficient convolutional neural networks for mobile vision applications *CVPR, 2018.*

[11] IANDOLA, F. N., HAN, S., MOSKEWICZ, M. W., ASHRAF, K.,DALLY, W. J., AND KEUTZER, K.Alexnet-levelaccuracy with 50x fewer parameters and<0.5 mb modelsize *arXiv preprint arXiv:1602.07360, 2016*

[12] KRIZHEVSKY, A., SUTSKEVER, I., AND HINTON, G. E. ImageNet classificationwith deep convolutional neural networks. *In NIPS, pp. 1106-1114, 2012*

[13] KRIZHEVSKY, A. AND HINTON, G. Learning multiple layers offeatures from tiny images *Technical Report, 2009.*

[14] YANN LECUN, CORINNA CORTES, AND CHRISTOPHER J. C. BURGES The MNIST database of handwritten digits *http://yann.lecun.com/exdb/mnist/, 1994. MNIST was created in 1994 andreleased in 1998.*

[15] Y. LECUN, L. BOTTOU, Y. BENGIO, AND P. HAFINER Gradient-based learning applied to document recognition *Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, November 1998.*

[16] NGUYEN, T. H., GRISHMAN, R.Relation Extraction: Perspective fromConvolutional Neural Networks *Workshop on Vector Modeling for NLP,pp. 39-48, 2015.*

[17] NVIDIA NVIDIA CUDA *Reference Manual, vol. 2.3, NVIDIA, 2009., http://developer.download.nvidia.com/compute/cuda/2_3/toolkit/docs/NVIDIA_CUDA_Programming_Guide_2.*

[18] RUSSEL, A. What, Exactly, Makes Something A Progressive Web App? *https://infrequently.org/2016/09/what-exactly-makes-somethinga-progressive-web-app/. (2016).*

[19] RUSSEL, A., SONG, J., ARCHIBALD, J. AND KRUISSELBRINK, M. Service Workers 1 *Editor's Draft, 22 December 2017. w3c.*

[20] SAINATH, T., MOHAMED, A.-R., KINGSBURY, B. AND RAMABHADRAN, B Deep convolutional neural networks for LVCSR *In Proc. Acoustics, Speechand Signal Processing 8614-8618, 2013.*

[21] SANDLER, M., HOWARD, A., ZHU, M., ZHMOGINOV, A., AND CHEN, L.-C Mobilenetv2: Inverted residuals and linear bottlenecks*CVPR, 2018.*

[22] SERMANET, P., EIGEN, D., ZHANG, X., MATHIEU, M., FERGUS, R., LECUN. Y. Oververfeat: Integrated recognition, localization and detection using convolutional networks*rXiv preprint-arXiv:1312.6229, 2013*

[23] STEINER, T. What is in a Web View? An Analysis of Progressive Web App Features When the Means of Web Access is not a Web Browser *WWW '18: Companion Proceedings of the The Web Conference 2018April 2018 Pages 789–796https://doi.org/10.1145/3184558.3188742*

[24] PATRICE Y. SIMARD, D., KRAUS,S., AND JOHN C. PLATT Best practices for convolutional neural networks applied to visual document analysis *Seventh International Conference on Document Analysis and Recognition, 2003, pp. 958–963*

[25] MA, N., ZHANG, X., ZHENG, H.-T., AND SUN, J. Shufflenet v2:Practical guidelines for efficient cnn architecture design. *ECCV, 2018*

[26] TAN, MINGXING, LE, QUOC V.EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks *arXiv:1905.11946v3*

[27] YANG, T.-J., HOWARD, A., CHEN, B., ZHANG, X., GO, A., SZE,V., AND ADAM, H.Netadapt: Platform-aware neural net-work adaptation for mobile applications *ECCV, 2018*

[28] ZEILER, M. D. AND FERGUS, R. Visualizing and understanding convolutional networks *CoRR, abs/1311.2901, 2013. Published in Proc. ECCV, pp.818-833, 2014*

[29] ZENG, D., LIU, K., SIWEI LAI, S., ZHOU, G. AND ZHAO, J. Relation classification via convolutional via convolutional deep neural network *In Proceedings of COLING 2014, pages 2335-2344, August 2014*

[30] ZHANG, X., ZHAO, J., LECUN, Y.Character-level Convolutional Networksfor Text Classification *arXiv: 1509.01626, 2015*

[31] ZHANG, X., ZHOU, X., LIN, M., AND SUN, J. An extremely efficient convolutional neural network for mobiledevices *CVPR, 2018*