To appear in A. Costa, V. Julian and P. Novais, eds. (2017):
Personal Assistants: Emerging Computational Technologies. Springer International

Personalized Visual Recognition via Wearables: A First Step toward Personal Perception Enhancement

Hosub Lee[1], Cameron Upright[2], Steven Eliuk[2] and Alfred Kobsa[1]

1: University of California, Irvine, 2: Samsung Research America

Abstract During the last few years, deep learning has led to an astonishing advancement in visual recognition. Computers now reach *near-human* accuracy in visually recognizing characters, physical objects and human faces. This will certainly allow us to build more intelligent personal assistants that can help users better understand their surrounding environments. However, most visual recognition systems have been designed for user-independent recognition (e.g., Google reverse image search), and not for an individual user. We believe this practice is restricting the technology from helping people who have individual needs. For example, a person with memory problems may want to have a computer that accurately recognizes a few close friends, rather than hundreds of celebrities. To address this issue, we propose a novel wearable system that enables users to create their own visual recognition system with minimal effort. A client running on Google Glass collects images of objects a user is interested in, and sends them to the server with a request for a specific machine learning task: training or classification. The server performs deep learning according to the request and returns the result to Glass. Regarding the training task, our system not only aims to build deep learning models with user generated image data, but also to update the models whenever new data is added by the user. Experiments show that our system is able to train the custom deep learning models in an efficient manner, in terms of the required amount of computing power and training data. Based on the customized deep learning model, the system classifies an image into one of 10 different user-defined categories with 97% accuracy.

Keywords Personalization; Person identification; Object recognition; Google Glass; Deep convolutional neural networks; Transfer learning; Finetuning

Introduction

With recent technological advances in wearable computing, users are more likely to collect information about their surroundings. For instance, users could directly capture images of objects through a smart glass rather than a smartphone. Google Glass is a representative wearable that can translate this scenario into reality. Since the camera functionality of Google Glass is always ready to be activated instantaneously by the user's command, it is reasonable to assume that more image data representing users' personal interests could be collected. Consequently, there are more chances to build a user-tailored visual recognition system by utilizing those collected images as training data.

Recently, deep learning has been making a breakthrough in diverse computer vision and pattern recognition problems [12, 16]. Deep learning is a machine learning technique that attempts to extract high-level concepts from data via a complex model composed of hierarchical processing units. The trained deep learning model then utilizes the extracted concepts in making predictions on new data. Deep convolutional neural networks (CNNs), a commonly used type of network, have been widely used in the computer vision community [25]. CNNs are biologically-inspired variants of artificial neural networks, which mimic how the human brain perceives images. These networks consist of multiple layers of filters which hierarchically process segments of the input image. Pooling layers are often added to reduce dimensionality and add translational invariance. Finally, multiple fully connected layers may be used to combine the spatial features and produce a final classification. Specifically, outputs of convolutions in the lower layers are used to represent the primitive element that forms the image (e.g., edge). Then, these representations are integrated in the higher layers to express more abstract concepts (e.g., shape) of the image. With this architecture, we can train the whole network through the standard backpropagation algorithm by using the labeled images as training data. Recent studies proved that CNN-based image classifiers have reached near-human levels on diverse visual recognition tasks [1, 11, 17, 19].

These technological advances can potentially benefit people. However, most deep learning applications thus far have been designed and developed for the general population, not for an individual who has a special need. Imagine that there is a professor who gives a lecture to 300 students. This professor might want to wear Google Glass displaying the names of students during the lecture because it is difficult to memorize all their names. People with visual impairments may want to have Google Glass proactively inform them about the presence of nearby friends. Those with cognitive impairment (e.g., dementia) could use a Google Glass capable of recognizing their personal belongings. This would help when they have memory problems. To realize all of these scenarios, each individual user needs to have a custom machine learning model trained on her/his own image data (i.e., a personalized machine learning model), and utilize the model for recognizing an input image. In this regard, we have defined two main requirements for constructing personalized machine learning models. First, a user should be able to easily collect images with appropriate labels in everyday life. Second, the system should be able to not only train a machine learning model with user generated image data, but update the model whenever the user provides the system with new classes of data.

With these requirements in mind, we developed a novel wearable system that enables users to create their own CNNs without any difficulties. To begin with, our system adopts the first generation of Google Glass for collecting image data representing users' personal interests. By using our Google Glass application named DeepEye, a user can take images of an object of interest and apply whatever label they want. DeepEye then transmits these labeled images to a GPU-equipped Linux server to train the CNN. In general, training deep learning models like a CNN from scratch uses a considerable amount of time on modern GPUs and requires very large volumes of training data. To make the training process more practical, we devised a new training mechanism named chained finetuning. This mechanism was designed to train a new CNN by utilizing the previously trained CNN as a starting point. Experimental results show that chained finetuning allows us to train the CNN while requiring less computational power and training data, compared to conventional training approaches. Most importantly, chained finetuning is effective for expanding the expressive power (i.e., the number of classifiable categories) of the CNN whenever the user collects images of new object classes. DeepEye can also run as an image classifier: if the user asks it to recognize an object from an image, it will show a classification result produced by the server. The server thereby utilizes the CNN trained specifically for the user. Our system showed about 97% accuracy in classifying an image taken by Google Glass into one of 10 user-defined categories.

In summary, the contributions of our work to the field of personal assistants are the following:

- We built a novel wearable system which lets users create their own deep learning-based visual recognition systems without any expertise;
- We proposed a simple, but efficient mechanism for training personalized deep learning models with user-generated image data (chained finetuning); and
- We showed the feasibility of the proposed system including chained finetuning through several visual recognition experiments.

Related Work

In the late 1990s, there were several attempts to build visual recognition systems into early versions of wearable computers. Steve Mann designed and prototyped a wearable personal device that could take pictures and recognize human faces in it [14]. This wearable device was also equipped with a small head-mounted display to give textual information to users. The author stated that the system could act as a visual perception enhancer because it could provide users with real-time feedback on what they were viewing. Even though this prototype was cumbersome to wear (e.g., a set of communication units were attached to the user's body), it is considered as a pioneering example of wearable visual recognition systems. Thad Starner *et al* proposed a system that recognizes the user's current behavior by analyzing video data [21]. The system utilized a hat-mounted camera to collect video streams, and then classified each single frame into pre-defined categories using a probabilistic object recognition technique. By using the results of object recognition, the authors trained a hidden Markov model (HMM) to identify three different tasks performed by the user. They also developed a visual recognition system capable of recognizing sentence-level American Sign Language selected from a 40-word lexicon [22]. They collected input video streams from both a desktop computer and from a wearable computer (namely the same device as in [21]). The experimental results showed that the system could recognize the given sign language subset with up to 98% accuracy. Finally, Antonio Torralba *et al* also proposed a wearable system that accurately identified 24 different of object types in a given image [23]. First, the authors adopted an HMM approach to recognize the current location of the user. Next, they utilized the location information as a contextual cue for detecting objects from an image, based on Bayesian inference. A helmet-mounted webcam was used to collect training image data under realistic conditions in which the user walked freely around the environment.

All of the mentioned works provide useful insight and practical advice for developing visual recognition systems for wearable computers. However, this topic has not been actively studied anymore after the early 2000s. This is probably because there were no commercial camera-equipped wearables available, leading to less opportunities for research in both academia and industry. However, with the advent of Google Glass, this situation may change. For instance, researchers at Fraunhofer developed emotion recognition software for Google Glass [3]. Based on their proprietary machine learning framework SHORE, the system detects people's faces in an image taken by Google Glass, and determines their emotional states by analyzing facial expressions. Similarly, researchers in the field of affective computing connected Google Glass with custom-made smile detection software to provide users with a real-time visualization of smiling faces of people around them [4]. Thomas Way *et al* designed a Google Glass application named ELEPHANT for retrieving meta information about the context (e.g., activity information) in which a picture was taken [24]. They anticipated that ELEPHANT could help people with memory impairment because it can provide contextual information when they have difficulty remembering a specific object. The authors consider using traditional machine learning algorithms such as logistic regression, support vector machines (SVMs) or Naïve Bayes to retrieve context information from an image.

Recently, researchers are trying to apply deep learning methods to wearable computers to achieve more accuracy in visual recognition systems. Recently, several companies demonstrated image classification with Google Glass [5, 18]. In order to recognize objects in an image captured by Google Glass, both utilized pre-trained deep neural networks which are deployed in their cloud. Since these works have not been published as yet, we do not know the details of their systems. However, it seems clear that they are focused on the classification of an input image using pre-trained deep learning models, rather than on training a deep learning model for an individual user.

Personalized Visual Recognition System via Google Glass

In this section, we discuss the design and implementation of our system in detail. We first describe an overall system architecture including software/hardware specifications, and then explain the functional details of the system.

System Architecture

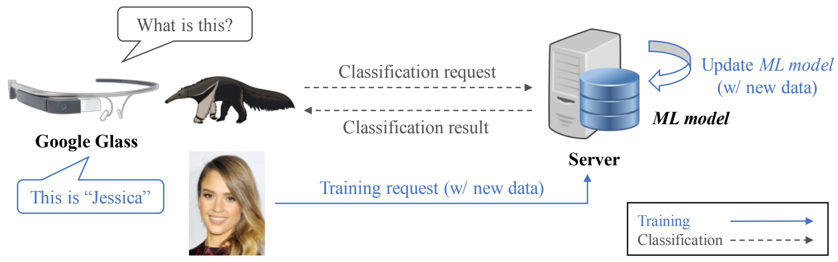


Fig. 1 System Architecture

Our system is designed as a client-server model (Fig. 1). As a client, Google Glass collects images when instructed by the user, and sends them to the server with a specific task type (training or classification). The server then carries out the requested task and returns the results back to Glass. The server was designed to continuously train (or update) the CNN using the proposed training mechanism whenever new image data is collected by Glass. When the server completes the training task, it replaces the preexisting CNN with the newly trained one that considers the most recent images. Overall, Google Glass acts as an image collector and interface which is visible to the end user. The server performs machine learning tasks in the background, such as classifying images when needed and training new models when an object is added. We chose this architecture because Google Glass has limited computing power. To the best of our knowledge, Glass's dual-core CPU (OMAP4430) and 2GB main memory are not sufficient to execute backpropagation for training CNNs.

Client

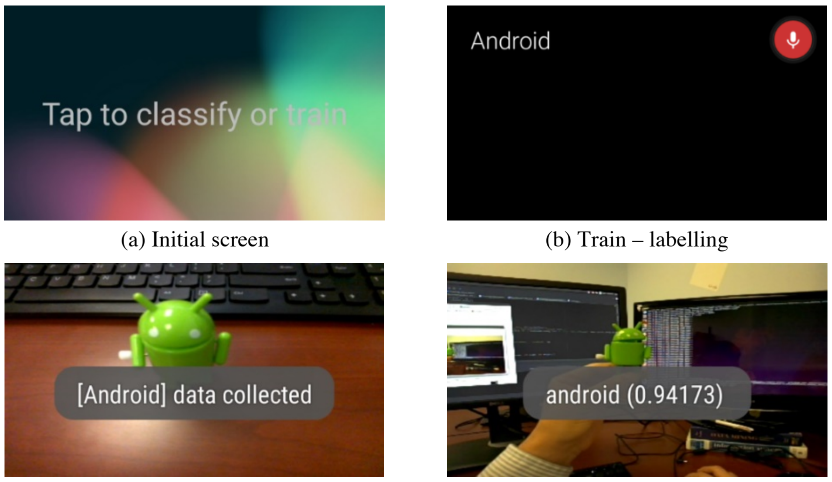
We developed a Google Glass application (Glassware) named DeepEye, following the Ongoing Task design pattern proposed by Google. The Ongoing Task pattern is commonly used for building a basic Glassware that enables users to control their Google Glass [2]. We wrote a function for DeepEye that takes a photo periodically upon the user's command. DeepEye sends these image data and messages to the server through Java socket communication over the Wi-Fi network. We used official Google libraries such as the Android 4.4.2 (API 19) SDK and the Glass Development Kit Preview in developing DeepEye.

Server

The main purpose of the server is to quickly train deep learning models with reasonable prediction accuracy. In order to achieve this, we built a Java server on a Linux workstation equipped with a modern GPU (NVIDIA GeForce GTX 970). We then deployed an open source deep learning framework named Caffe [9] on the server. Currently, Caffe is one of the fastest CNN implementations available. If the server receives a request for a specific task from DeepEye, it executes a corresponding Caffe command (e.g., train a CNN or classify an image with a CNN) through its python interface, and returns the result.

Workflow

As discussed earlier, DeepEye has two main tasks: training and classification. Here, we describe each task step by step. When DeepEye is started, a user is asked to choose between two tasks via the Google Glass touch pad (Fig. 2-a).



(a) Initial screen (b) Train – labelling

(c) Train – data collecting (d) Classification

Fig. 2 Screenshots of DeepEye

Training

For the training task, the user enters the name of the target object (i.e., its label) through Google Voice Input (Fig. 2-b). The user can try again if the result of the speech recognition was incorrect. When the user confirms the label, DeepEye begins to take a photo of the object every five seconds, and transmits it with a message representing the current task (_train) to the server. This process is repeated as long as DeepEye receives an ACK message from the server and the user has not explicitly terminated the training task (Fig. 2-c). The server will use the transferred image data for training a deep learning model via our proposed training mechanism which we call ‘chained finetuning’.

As discussed, training deep learning models from scratch is very expensive and time-consuming. For example, training a CNN on the ImageNet dataset which contains 1.2 million images with 1,000 categories can take several weeks on a single GPU or hours/days in a distributed setting [11]. For these reasons, it is more common to retrain an already fully trained model on a new dataset to repurpose a preexisting model for different tasks [10]. For instance, after the initial retraining, we can immediately exploit the pre-trained CNN's well-learned parameters representing generic visual features like edges. Then, we can focus on updating values of parameters aimed at extracting more object-specific (high-level) features related to our own image data. This approach is known as finetuning, one kind of transfer learning algorithm. Finetuning is widely used to avoid expensive training efforts in diverse machine learning tasks [15].

Chained finetuning, the extended version of finetuning, was designed to train a new deep learning model on ad hoc additional training data. The main idea of chained finetuning is simple. To train a new model (here, CNN) for a new task, it iteratively retrains the pre-trained CNN on a newly created dataset. Suppose that there exists a CNN trained to classify an image into three user-defined categories A, B and C (CNN_ABC). If a user adds a new category D with the corresponding image data, chained finetuning then constructs a new model (CNN_ABCD) on new training data while using the old model (CNN_ABC) as a starting point. More specifically, we define a new CNN by adopting an underlying network architecture of the pre-trained CNN, but change its classification layer to have a correct number of outputs based on the given task (e.g., 4 output nodes for CNN_ABCD). Next, we can initialize parameters (weights) of the new CNN with that of the pre-trained CNN, and then progressively update the weights of the new CNN through the backpropagation algorithm on a new dataset. This process can be continued in a series whenever new types of training data became available.

Chained finetuning begins if there are at least two user-defined categories with a sufficient amount of training data. Through repeated experiments, we determined the threshold for sufficient training data as 100 images per class. The process also checks whether there are any ongoing CNN training processes on the system. If training is already in progress, it will not try to train a new model until the ongoing process has ended. Otherwise, if this is the first finetuning attempt, it trains a new model by using the pre-trained CNN named BVLC Reference CaffeNet (CaffeNet) [9]. We utilized CaffeNet as a base model because it is a publicly available pre-trained CNN that has a reasonable prediction performance on a 1,000-class object recognition task (ImageNet challenge) [11]. In any later finetuning, it trains a new model by finetuning the CNN pre-trained in the previous finetuning stage. When a finetuning process has finished, the previously trained CNN is replaced with the newly trained CNN.

Classification

Classification is relatively simple. When users choose the classification task, they take a picture of the object by clicking the Google Glass touch pad. Similar to the training task, DeepEye sends the image to the server, but with a different message (_classify).

Next, the server uses the latest trained CNN to execute the Caffe classification command on the image. If no error occurs, the server sends the classification result (with probability) back to DeepEye. If DeepEye receives the result from the server, it displays them to the user through Google Glass's heads-up display (Fig. 2-d).

Experiment 1: Person Identification

Overview

To validate the effectiveness of the chained finetuning mechanism, we designed and conducted a series of person identification experiments. At first, we finetuned CaffeNet so that it could identify 20 different people, rather than 1,000 different objects from a set of images. The intention was to confirm that finetuning is an effective approach for constructing a custom deep learning model for a new task. This was important because a single finetuning step is the basic building block for chained finetuning. Second, we finetuned the previously trained CNN while adding images of a new person to the training data (i.e., chained finetuning), and evaluated the predictive power of the CNN trained in each single finetuning stage. As a result, we trained a custom CNN so that it could classify five different people. While finetuning CNNs, we tried to update the weights of the classification layer faster than that of the underlying (low-level) layers. This is because low-level layers of CNNs are supposed to extract more generic visual features (e.g., edges), and therefore they likely do not change much when presented with new data. Higher layers, in contrast, represent more class-specific characteristics (e.g., shapes) and thus need major updating with new data. Finally, we compared chained finetuning with the original finetuning approach to decide which is better for training personalized deep learning models.

Training Data

To gather training data, we randomly downloaded photos of 20 celebrities via Google Image Search. Using a simple shell script, we collected a maximum of 100 images for each person. We excluded some duplicate or corrupt images, and hence the number of images per class (person) was not the same (see Table 1). We cropped faces from original images using the OpenCV library to better gauge how well the trained CNNs identify different faces. We also augmented training data by creating additional image transformations using ImageMagick's 'convert' tool. Specifically, we created four variations of each original image through 90, 180 and 270-degree rotation and mirroring. We included this step to alleviate potential overfitting problems as much as possible by providing more training data without extra labelling cost (data augmentation [7, 11]). In total, our training data included 6,220 images. To measure training and test errors of the trained CNNs, we shuffled the training data and put 20% aside as test data.

Table 1 Person Identification – Training Data

No	Label	Number of Images[1]	Characteristics
1	Jessica Alba	68 (340)	Female, 30s
2	Kate Upton	54 (270)	Female, 20s
3	Scarlett Johansson	67 (335)	Female, 30s
4	Emma Watson	73 (365)	Female, 20s
5	Jennifer Lawrence	60 (300)	Female, 20s
6	Arnold Schwarzenegger	49 (245)	Male, 60s
7	Johnny Depp	63 (315)	Male, 50s
8	Bill Gates	59 (295)	Male, 60s
9	Kristen Stewart	80 (400)	Female, 20s
10	Leonardo Dicaprio	81 (405)	Male, 40s
11	Lionel Messi	55 (275)	Male, 20s
12	Manny Pacquiao	51 (255)	Male, 30s
13	Matt Damon	74 (370)	Male, 40s
14	Michael Jackson	47 (235)	Male, 50s
15	Sandra Bullock	75 (375)	Female, 50s
16	Eminem	39 (195)	Male, 40s
17	Steve Jobs	55 (275)	Male, 50s
18	Tiger Woods	58 (290)	Male, 40s
19	Tom Cruise	74 (370)	Male, 50s
20	Will Smith	62 (310)	Male, 40s

[1] The number in parentheses indicates the number of augmented training images.

Finetuning for 20-Class Person Identification

We finetuned CaffeNet for identifying 20 different face photos. By using all images described in Table 1 as training data, we updated all the weights in CaffeNet via back-propagation, with a maximum of 5,000 iterations. The training curves depicted in Fig. 3 show that the finetuned CaffeNet started to converge around the 1,000th iteration. In our training/test data set, we could not observe any serious overfitting as both training and test error show a similar pattern during the entire training process. For 40 consecutive tests on the 20% test data set, its average prediction accuracy was about 0.98 and its loss (error) was about 0.05. We therefore conclude that finetuning is effective for transforming a preexisting deep learning model into the new model that performs a different task.

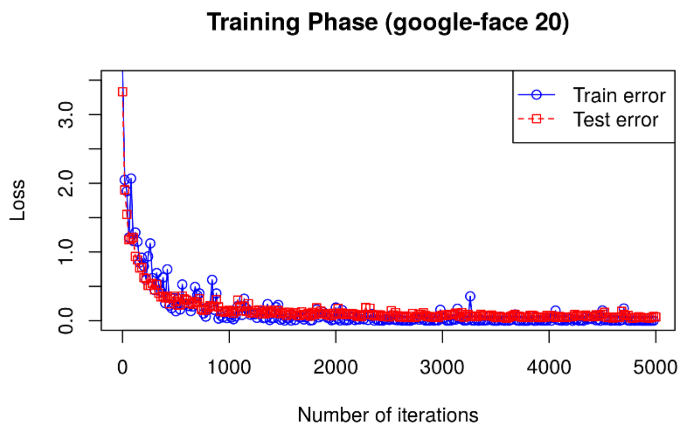


Fig. 3 20-Class ‘Person’ Identification – Training Curves

Chained Finetuning for 5-Class Person Identification

The goal of this experiment was to train a new CNN that identifies 5 different people through chained finetuning. The experiment was conducted in the following steps. First, we finetuned CaffeNet on training images of class 1-3 so as to identify 3 different faces, and used it as a base model for chained finetuning. Next, we continued to finetune the previously trained CNN whenever a new data class was added. Two additional classes of image data were added in turn to the previous training data (class ‘new-1’ and ‘new-

2’ in Table 2). There are 1,625 images in the training data. We shuffled and split them into 80% training and 20% test data.

Table 2 5-Class ‘Person’ Identification – Training Data

No	Label	Number of Images	Characteristics
1	Jessica Alba	68 (340)	Female, 30s
2	Kate Upton	54 (270)	Female, 20s
3	Scarlett Johansson	67 (335)	Female, 30s
new-1	Alexandra Daddario	70 (350)	Female, 30s
new-2	Amanda Seyfried	66 (330)	Female, 30s

Since we noticed in the previous experiment that the finetuned network converged around the 1,000th iteration, we decided to stop our individual finetuning at this point. Table 3 summarizes the prediction performance of the chain-finetuned CNNs (CF_CNN) on the 20% test data set. Similar to the previous experiment, the test accuracy of the finally trained CNN was nearly perfect (99%). In addition, all CNNs trained through the chained finetuning mechanism also showed promising test accuracies. Fig. 4 displays training curves for the finetuned model, CF_CNN (5). As with the previous experiment, no serious overfitting on the training/test data sets was observed.

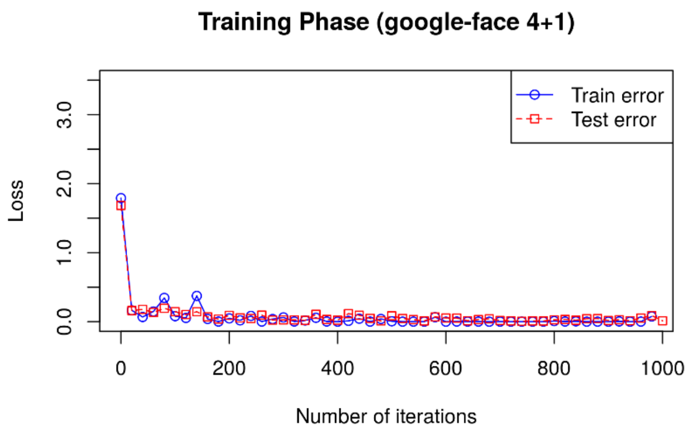


Fig. 4 5-Class ‘Person’ Identification – Training Curves

Table 3 5-Class ‘Person’ Identification – Test Accuracy

Finetuned Model (Number of Classes)	**Base Model (Number of Classes)**	**Test Accuracy (Loss)**
CF_CNN (3)	CaffeNet (1,000)	0.9885 (0.0427)
CF_CNN (4)	CF_CNN (3)	0.9956 (0.0278)
CF_CNN (5)	CF_CNN (4)	0.9969 (0.0134)

Comparison between Finetuning and Chained Finetuning

One possible approach to cope with an ad hoc addition of a new data class is to train a new CNN using CaffeNet as a *fixed* base model whenever new data is added, which is the original finetuning approach. To compare this approach with chained finetuning, we used original finetuning in training a CNN on training data used in the previous experiment (finetuned CNN; F_CNN). Then, we compared its prediction power with the CNN trained through chained finetuning (chain-finetuned CNN; CF_CNN). To gauge the models’ prediction power more objectively, we collected an additional set of 50 images per each class. These images were downloaded from a different source (Bing image search) and never used in the training process. We used them as a validation data set for this experiment.

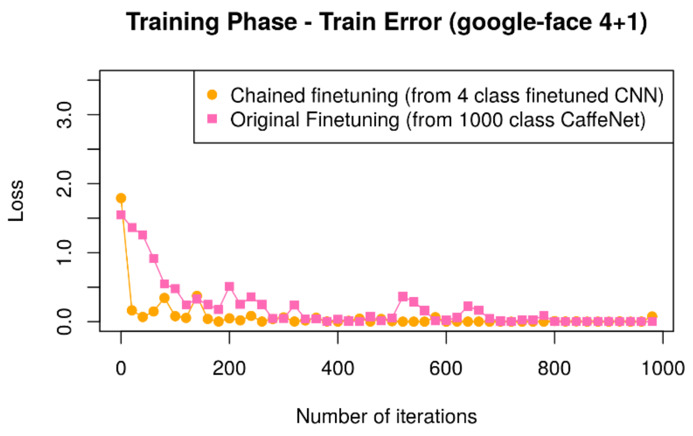


Fig. 5 5-Class ‘Person’ Identification – Finetuning vs. Chained Finetuning (Training Error)

As shown in Fig. 5, the chain-finetuned CNN starts to converge about 30% earlier (after 200 iterations) than the finetuned CNN (after 700 iterations). This was expected, since chained finetuning takes advantage of what was already learned from the previous step. On both the test and validation data sets, the chain-finetuned CNN outperformed the finetuned CNN (see Fig. 6 and Table 4). However, we also noticed that the performance on validation data (validation accuracy) was lower than the test accuracy in both cases. This implies that the trained CNNs might be excessively fitted to the training data, thus having difficulties to predict outcomes for previously unseen data. We suspect that the unbalanced distribution of training data is one possible reason for this overfitting problem. There were 400 training images for class 9, but 195 images for class 16 (see Table 1). The model may not have been sufficiently trained for identifying class 16. For the following experiments, we tried to assign an equal amount of training data to each individual class to prevent overfitting as much as possible.

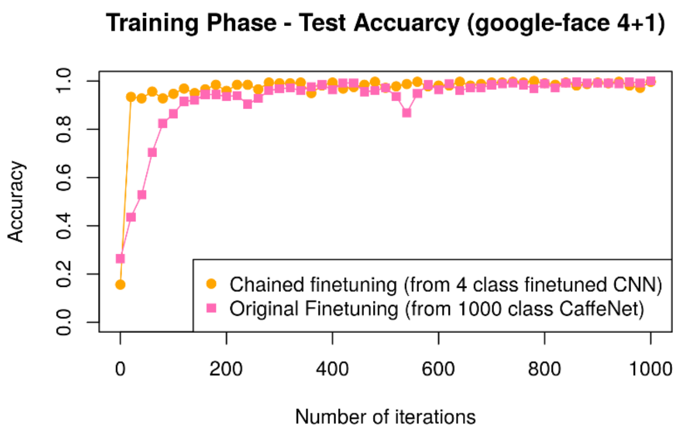


Fig. 6 5-Class 'Person' Identification – Finetuning vs. Chained Finetuning (Test Accuracy)

Table 4 5-Class 'Person' Identification – Test and Validation Accuracy

Finetuned Model (Number of Classes)	**Test Accuracy (Loss)**	**Validation Accuracy (Loss)**
F_CNN (5)	0.9656 (0.0893)	0.844 (0.7122)
CF_CNN (5)	0.9969 (0.0134)	0.88 (0.6967)

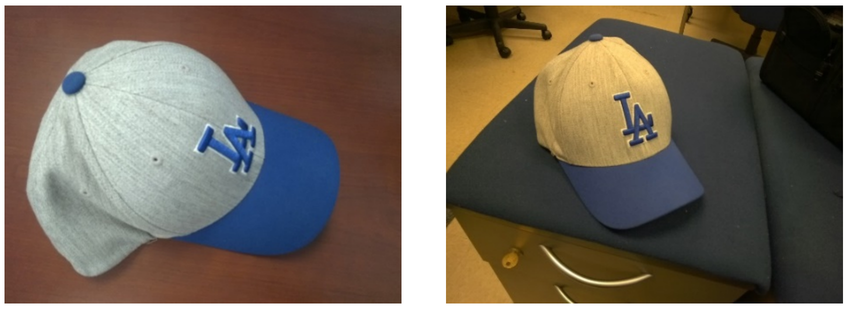
Experiment 2: Object Recognition

Overview

In this experiment, we aimed to evaluate the predictive power of chain-finetuned CNNs in a real-world scenario. To this end, we trained a CNN so that it can recognize 10 different types of objects from images taken by Google Glass. The ultimate aim of such a system would be to help people with memory problems to remember and recognize their personal belongings.

Training and Validation Data

To begin with, we chose 10 personal objects (small toy, badge, baseball cap, key, eyeglasses, pouch, food container, lotion, watch, wallet) of a member of our research team, and collected images using DeepEye and the server. To minimize the risk of overfitting, we collected the exact same amount of training data for each class, namely 100 original with 400 automatically augmented images. We also collected 30 additional images per each class as validation data. To differentiate these from the original training data, we deliberately varied the photographing conditions such as lighting, angle and background (see Fig. 7). Both training and validation images were taken by a single participant in a standard office setting. Even though Google Glass is equipped with a 5MP camera capable of taking 2,560 by 1,888 resolution JPG images with a file size of about 2 megabytes, we collected reduced-size versions of the images (1296 by 972 pixels) to avoid any network delays between DeepEye and the server.



(a) Sample Image for Training (cap) (b) Sample Image for Validation (cap)

Fig. 7 10-Class 'Object' Recognition – Training and Validation Data

Chained Finetuning for 10-Class Object Recognition

Regarding chained finetuning, we used the same procedures and settings as for the 5-class person identification experiment described above. The training curves in Fig. 8 show that the finetuned CNN starts to rapidly converge at around 100 iterations. Compared to all previous person identification experiments, this one had near-perfect test accuracy, probably because the classification task was easier. The objects used in this experiment had vastly different shapes and appearances (e.g., cap vs. wallet), so that the model could identify them with high confidence. In contrast, the differences between faces of the same gender and age are subtle (e.g., class 4: Emma Watson and class 9: Kristen Stewart in Table 1). This could have led to some confusion telling them apart.

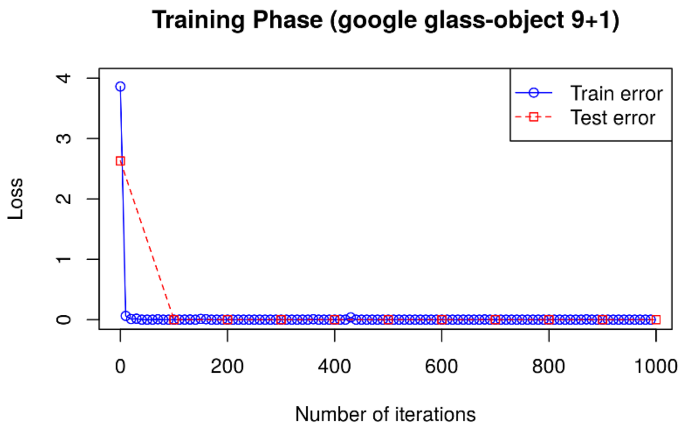


Fig. 8 10-Class 'Object' Recognition – Training Curves

Table 5 summarizes the measured prediction power of all chain-finetuned CNNs on the validation data set. For up to 7 different objects, the trained CNNs showed a near-perfect performance in recognizing objects without serious overfitting concerns. However, the validation accuracy was slightly diminished, as the number of object classes increased from 8 to 9. It may be improved if we collect additional training images of the objects, and train a new CNN at the next finetuning stage. The validation accuracy of the final trained CNN was 97%, with a loss of 0.116. It took approximately 7 minutes to train this model on our GPU platform.

Table 5 10-Class 'Object' Recognition – Validation Accuracy

Finetuned Model (Number of Classes)	Base Model (Number of Classes)	Validation Accuracy (Loss)
CF_CNN (3)	CaffeNet (1,000)	0.99 (0.0212)
CF_CNN (4)	CF_CNN (3)	0.99 (0.1819)
CF_CNN (5)	CF_CNN (4)	0.99 (0.0555)
CF_CNN (6)	CF_CNN (5)	0.99 (0.0462)
CF_CNN (7)	CF_CNN (6)	0.99 (0.0454)
CF_CNN (8)	CF_CNN (7)	0.96 (0.2696)
CF_CNN (9)	CF_CNN (8)	0.94 (0.2319)
CF_CNN (10)	CF_CNN (9)	0.97 (0.116)

Discussion and Future Work

We demonstrated the feasibility of our proposed visual recognition system that uses Google Glass. Yet, some issues still need to be overcome before it can be used more widely.

Google Glass emits a lot of heat when it continuously utilizes the camera function. According to [13], a single camera usage heats Google Glass 28°C above the surrounding temperature. In the worst case (video chatting), Google Glass's surface temperature increased up to 50°C within 13.3 minutes. Because Google Glass is in direct contact with the skin, the heated surface may lead to discomfort and potentially even health risks for users. Therefore, users may have trouble collecting at once a large volume of images (more than 100) via Google Glass. The authors also measured the energy consumption of Google Glass for various tasks. To take a single photo, Google Glass consumes 2,927mW in 3.3 seconds. Users can take fewer than 800 images on a single battery charge. Like the heat problem, this may prevent users from taking sufficiently many images to build their own deep learning models. We expect Google to fix these issues in the next generation of Glass.

At this moment, there exists no large-scale image dataset collected from wearable computers such as Google Glass. Therefore, we generated the custom dataset using DeepEye in our experiments, and utilized it for testing the proposed training mechanism. To thoroughly verify its effectiveness, we should still investigate whether our mechanism also works well for more complex image classification problems (e.g., 100-class object recognition). Thus, we are considering distributing DeepEye to a group of Google Glass users, and to collect abundant image data from their everyday lives. Furthermore, we need to tackle any potential overfitting problems in training personalized deep learning models. For this, we applied a neural network regularization technique named dropout [20] to DeepEye's training mechanism. Dropout forces neural networks to learn several independent representations of identical input-output pairs, by randomly disabling some neurons (nodes) in a given layer. For all experiments described above, we used a fixed dropout rate of 0.7 for fully-connected layers. Therefore, it is worth investigating the optimum dropout rate for training more complex models through chained finetuning.

While often overlooked, privacy is an important concern [6, 8]. For the person identification task, users need to take photos of people around them (mostly, friends and acquaintances). We assumed that they would ask them for their permission before taking a photo. However, there are no user interfaces or mechanisms in our system advising them to do that. The system may invade privacy if it collects photos of people without their consent. To find the best way to prevent possible privacy invasions, we need to collect users' opinions on, and/or reactions against the system. Also, it is necessary to prevent unauthorized access to user-generated image data and the trained models, as they may reflect a user's very personal behavior and interests.

Finally, we believe that even users who are not tech-savvy should have little difficulty using our system because they are only asked to perform a few simple operations via Google Glass (e.g., image labelling through Google Voice Input). However, we need to verify the usability of the system with target users who have special needs. Specifically, we need to qualitatively and quantitatively assess the usability of the system for people with memory or visual impairments, possibly including their caregivers. Their feedback may allow us to improve our user interface, so that our system will work in a more user-friendly way. Additionally, a longitudinal study might be needed to verify whether our system can have a positive influence on their lives and medical conditions.

Conclusion

In this paper, we designed and implemented a novel wearable system which builds personalized deep learning models for recognizing objects of interest to a user. To the best of our knowledge, this is the first attempt to train deep learning models for *personalized* visual recognition, via camera-equipped wearable computers like Google Glass. The proposed system works as a client-server model: Google Glass (client) collects images from a user's everyday life and sends them to a GPU-equipped Linux server. The server then trains a deep convolutional neural network (CNN) on the user-specific image data. To efficiently update the pre-trained network on newly-added images, we proposed a simple training mechanism called chained finetuning. As a variant of conventional finetuning, it is effective in terms of prediction power and training efforts in continuously training (or updating) a personalized deep learning model. In a custom 10-class object recognition task, our system took 7 minutes to train a personalized CNN on our GPU platform, and showed a 97% classification accuracy without serious overfitting. Considering the training time and the model's prediction power, we believe our system can become a feasible intelligent personal assistant. Future work will mainly focus on the testing of the proposed system with more users and harder tasks. It will also include privacy impact assessments and a verification of its effectiveness in improving users' cognitive abilities.

Acknowledgements

Part of this work was done while Hosub Lee was a summer intern at Samsung Research America, Mountain View, CA. We would like to thank our reviewers for their valuable comments on earlier versions of this paper.

References

1. Ciresan D, Meier U, Schmidhuber J (2012) Multi-column deep neural networks for image classification. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, IEEE, pp 3642–3649
2. Google Developers (2015) Ongoing task pattern. https://developers.google.com/glass/develop/patterns/ongoing-task
3. Fraunhofer (2014) Fraunhofer IIS presents world's first emotion detection app on Google Glass. http://www.iis.fraunhofer.de/en/pr/2014/20140827_BS_Shore_Google_Glas.html
4. Hernandez J, Picard RW (2014) SenseGlass: using Google Glass to sense daily emotions. In: Proceedings of the adjunct publication of the 27th annual ACM symposium on user interface software and technology, ACM, pp 77–78
5. Hof R (2014) First image recognition app coming soon to glass. http://www.forbes.com/sites/roberthof/2014/02/26/first-image-recognition-app-coming-soon-to-google-glass/
6. Hong J (2013) Considering privacy issues in the context of Google Glass. Commun. ACM, 56(11):10-11
7. Howard AG (2013) Some improvements on deep convolutional neural network based image classification. arXiv preprint arXiv:1312.5402

8. Hoyle R, Templeman R, Armes S, Anthony D, Crandall D, Kapadia A (2014) Privacy behaviors of lifeloggers using wearable cameras. In: Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing, ACM, pp 571–582
9. Jia Y, Shelhamer E, Donahue J, Karayev S, Long J, Girshick R, Guadarrama S, Darrell T (2014) Caffe: Convolutional architecture for fast feature embedding. In: Proceedings of the 22nd ACM International Conference on Multimedia, ACM, pp 675–678
10. Karayev S, Trentacoste M, Han H, Agarwala A, Darrell T, Hertzmann A, Winnemoeller H (2013) Recognizing image style. arXiv preprint arXiv:1311.3715
11. Krizhevsky A, Sutskever I, Hinton GE (2012) ImageNet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp 1097–1105
12. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521:436–444. doi: 10.1038/nature14539
13. LiKamWa R, Wang Z, Carroll A, Lin FX, Zhong L (2014) Draining our glass: an energy and heat characterization of Google Glass. In: Proceedings of 5th Asia-Pacific Workshop on Systems, ACM, p 10
14. Mann S (1997) Wearable computing: a first step toward personal imaging. Computer 30:25–32. doi: 10.1109/2.566147
15. Pan SJ, Yang Q (2010) A survey on transfer learning. Knowledge and Data Engineering, IEEE Transactions on 22:1345–1359. doi: 10.1109/TKDE.2009.191
16. Schmidhuber J (2014) Deep learning in neural networks: an overview. Neural Networks 61:85–117. doi: 10.1016/j.neunet.2014.09.003
17. Sermanet P, Eigen D, Zhang X, Mathieu M, Fergus R, LeCun Y (2013) OverFeat: Integrated recognition, localization and detection using convolutional networks. arXiv preprint arXiv:1312.6229
18. Simonite T (2013) A Google Glass app knows what you're looking at. http://www.technologyreview.com/view/519726/a-google-glass-app-knows-what-youre-looking-at/
19. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556
20. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research 15:1929–1958.
21. Starner T, Schiele B, Pentland A (1998) Visual contextual awareness in wearable computing. In: Wearable Computers, Second International Symposium on, IEEE, pp 50–57
22. Starner T, Weaver J, Pentland A (1998) Real-time American sign language recognition using desk and wearable computer based video. Pattern Analysis and Machine Intelligence, IEEE Transactions on 20:1371–1375. doi: 10.1109/34.735811
23. Torralba A, Murphy KP, Freeman WT, Rubin MA (2003) Context-based vision system for place and object recognition. In: Computer Vision, Ninth IEEE International Conference on, IEEE, pp 273–280

24. Way T, Bemiller, A, Mysari R, Reimers C (2015) Using Google Glass and machine learning to assist people with memory deficiencies. In: Proceedings on the International Conference on Artificial Intelligence (ICAI), pp 571–577
25. Zeiler MD, Fergus R (2013) Visualizing and Understanding Convolutional Networks. In: European Conference on Computer Vision, Springer International Publishing, pp 818–833