Stylist Assistant

Capstone Final Report 2018

Arjun Kalburgi

Table of Contents

Problem Domain

The Stylist Assistant is a companion for stylists and their clients. In the industry today, personal stylists work with their clients closely, often on a daily basis, helping them achieve the client's desired personal style goal. This makes being a personal stylist a full-time job, and hiring a personal stylist very expensive. Hence, participation in the industry is limited.

Stylist Assistant uses AI to help stylists with clients' daily style. Reducing the time commitment of the personal stylist reduces time and cost barriers for clients, thus increasing participation in the industry.

The project as it stands relies on stylists to help the app build reliably fashionable pieces tailored to the client's style goal. It is possible to have the system work without the stylist if a sufficiently large and complex database existed. However, featuring the stylist in the app allows more potential avenues of growth for the industry, similar to that of the personal training industry where trainers have their own sponsorship deals, or create their own products and communities. The barrier to entry for stylists and potential clients limits the personal stylist industry from this type of growth.

The project's potential to disrupt and grow the industry to a large degree makes it a potentially revolutionary technology.

Existing Solutions

The majority of mobile fashion technologies in the market today only act as a wardrobe manager or organizer, allowing for categorization and planning of outfits and items. Other fashion-related applications online offer virtual "try before you buy" features. These are all tools for users to essentially be their own personal stylist; there is no recommendation component in these apps.

In research, there have been a number of papers written about the relationship between AI and fashion, including the DeepFashion project from which our solution was kickstarted. A majority of these papers look into the topic mainly as an exploration and not as a solution to the opportunity presented in the previous section. This includes papers like "An intelligent clothes search system based on fashion styles" by Ching-I Cheng, which explores fashion search in images. Other papers, such as Haosha Wang's "Machine Fashion: An Artificial Intelligence Based Clothing Fashion Stylist", ask the question nearly identical to ours: can an AI machine be a Fashion Stylist?

Wang's project has an offshoot project called Style-Me which, like the Stylist Assistant, recommends fashion looks. However Style-Me uses fashion trends and users' personal style history to make recommendations and therefore does not sit in the same position in the industry as the Stylist Assistant. Additionally, Style-Me's purpose is to show that a machine can indeed be a fashion stylist, and not to actually enter the market. Similarly, most other offerings found online, although tools for recommending fashions, are moreso positioned to entice additional purchases from customers instead of looking to become impactful resources within the fashion and styling industry.

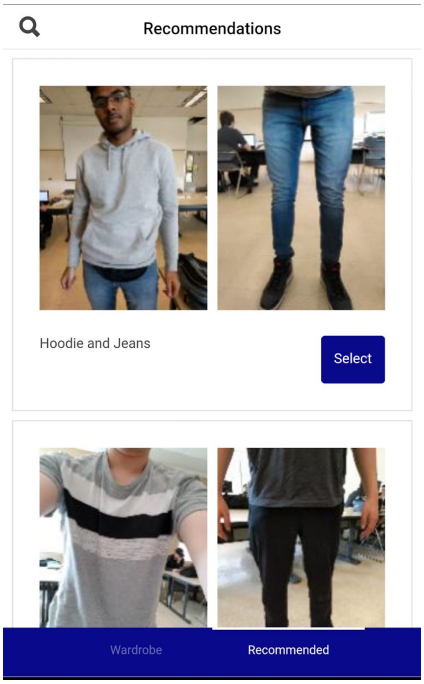
Design Choices

The Stylist Assistant is used by both stylists and clients, meaning a user can be both a stylist and a client using the same app (provided they have different email addresses). This design decision was made to help keep the project's scope relatively narrower. We noted that startups like Uber originally kept both types of users, rider and driver, on the same app, but then grew into two apps as both sides of the service expanded. If Stylist Assistant were to continue to grow, this would likely become a solution for us as well.

A two-application solution may actually be a more ideal solution for this industry as well. There is a possibility that the best way to enter Stylist Assistant into the market would be to build a unique and custom application for each stylist featuring the client side of the current Stylist Assistant. The stylist themselves could log into the application as it stands now with the client side stripped out (stylist only). This option is a business decision and would need further research to decide.

From a technical standpoint, the biggest design choice we made was to build our own neural net for the image tagging. We wanted to explore creating a custom classifier with the IBM Watson API, but decided

this route would not satisfy the workload and learning requirements for the course. In our quick test, using DeepFashion and the custom classifier worked well as the Watson API is assisted by its existing knowledge base. By Scott's recommendation, we settled with building our own neural net, using the TensorFlow library as a solution that worked and satisfied the course requirements.



Much thought was put into the visual/graphic branding, though not decided upon as a group. The visual aspects of the application, including the colours and fonts, were chosen to support emotional responses of reliability and elegance. The font, Eloquent Pro, and the colour black give an impression of sophistication. In colour theory, the use of navy is known to build feelings of trust and reliability. These visual cues are present in the application in subtle ways, such as the tab-bar colour of the recommendations being navy to suggest that the recommendations are trustworthy. (Pictured right). The rest of the application is a simple black and white to create that tone of elegance.

Design patterns (visual layouts of the app views) were given less detailed thought, although some minor patterns which were more intuitive were implemented , such as tabs in the client side rather than a side menu. This follows the general design rule that discards the idea of keeping main features hidden in a side-menu navigation. The stylist side does feature a side-menu, but it is a much more acceptable design pattern for switching between clients. Similarly, the display of side-by-side cards was chosen over other list options for showing the wardrobe and recommended outfits of the client.

Impacts

Societal Impacts

To estimate the societal impacts of the Stylist Assistant and the potential growth of the personal stylist industry, we can look at the personal fitness trainer industry, a 1-on-1 industry that has already shown expansive growth in society.

Personal trainers themselves can garner large social media influence and popularity. This helps them earn endorsement and product deals in addition to being more sought after as a personal trainer. However, this popularity contest can make the industry very tough and competitive, resulting in personal issues between trainers as well as shrewd business deals that can negatively affect members of the community. Both issues may occur if the personal stylist industry were to flourish, perhaps to an even greater degree as its parent industry, fashion, is already a shrewd industry.

The personal trainer industry benefits from having a wide market of consumers. As fashion has seen growing cultural significance over the last decade, the Stylist Assistant may influence trends in new ways. For example, having stylists choose outfits and wardrobe pieces for clients more frequently may

decrease the number of weaker trends that circulate, or perhaps trends will cycle through quicker as more people stay up-to-date with current trends.

Environmental Impacts

The app has very small environmental impacts. Our biggest factor with regards to a carbon footprint is the server, which hosts both the neural network and the HTTP server for the app. We could not retrieve exact data from Cybera on our energy usage. However, it is well known that training a neural network is energy intensive, and since Cybera is powered by coal, it will result in a large carbon footprint.

Stakeholder Impacts

- *Client (Dr. Scott Dick)* The client has the fun experience of marking a game.
- *Developers* The Stylist Assistant has had a positive impact on the developers, allowing us to learn and grow. We have had positive feedback from friends after demonstrations and presentations and are encouraged by the product's ability, knowing its limitations.
- *Judges* From feedback throughout the Capstone presentation, Design Showcase poster presentations and demonstrations, the project seemed to impress and engage the judges that visited and talked to us.
- *Stylists* The Stylist Assistant provides a positive impact for stylists by allowing them to preset daily styles for their clients, thereby freeing them to take on more clients and more projects. Additionally, societal impacts of the Stylist Assistant (discussed above) can be very beneficial for stylists who take advantage of them. It would also allow for an additional income stream to the many fashion influencers already on social media.
- *Stylist's Clients* The Stylist Assistant provides a positive impact on clients by offering them with a much more accessible personal stylist alternative. It helps achieve the desired daily style goal and all the other benefits of having a personal stylist with a relatively hands-off experience. The app provides the dream coaching experience.
- *Clothing brand/manufacturers* Thanks to the Stylist Assistant, clothing brands now have an additional option to market and sell their brand by supporting and endorsing stylists.
- *Competitors* As mentioned in the Existing Solutions section, we have been unable to find real competitors on the market that share our software vision of providing recommended outfits.

Roles

I took on 3 main roles in the project: a business analyst (BA) for my domain knowledge of the fashion industry, a visual designer (designer) for my contributions to create the branding of the application, and an application developer (dev) for my contributions in creating the mobile application.

Business Analyst

I assumed the role of BA because I had the original idea and vision for the project. My understanding of why the solution is important, as explained in Question 1, helped me lead team discussions on the project and consider real-world applications when making design decisions. I think these discussions were instrumental in creating a proper application.

Aside from leading design discussions, my role as BA included verifying the results of the neural network and similarity matrix for viability and correctness. At one point, the similarity matrix produced a 70% similarity for two items that in reality were very different. During my review of the results, I helped James determine why his algorithm was not working (incorrect attribute tagging), and led the group to shift their focus and rely more heavily on category tagging. When the neural network suffered from screwed data, I helped Vitor find a solution by trimming the list of categories in order to limit the variation in the samples.

Designer

I fell into the role of designer naturally due to my existing skill set and how well it translates from my role as BA. Using my understanding of the stylist industry as a BA, I applied the same principles to design the branding, assets and design patterns of the application as discussed in Question 3. I worked mostly individually here; James and Vitor did not have much input into the design choices I made. These were considered my tasks.

Application Developer

I took the layouts, assets and branding I created as a designer and applied or built off them as a dev. Though James also did some work on the front end of the app, that was mostly to initialize the views with Ionic's default templating. I would then redesign the views using CSS and HTML based on the mockups I designed.

Eventually James stopped working on the application in order to focus on the similarity matrix and the rest of the backend, leaving me as the sole application dev. During this time I did a large majority of the features and logic, including re-implementing Firebase registration and sign-up to feature proper rejection of unqualified users, and implementing Firebase in the rest of the app with dummy data (projectstarter.json) to fill content. The dummy dataset would grow with each feature I implemented, as I needed more and more data to support those features. The item filter feature required categories in each item object; the dress log required the date in each object; etc. Cycling between creating new dummy data and building out features to support them was how I completed much of the app.

James and I eventually worked together to finish out the last few features and bugs in the app. Implementing, uploading, downloading and removing images from the database, and re-implementing the outfit recommendations were the tasks that we took up in this role. As the two application developers, we did a great job working together.

James Wu

Implied from the description above, James took the role of full-stack developer, working both on the application and server sides of the project. In the server, James built the similarity matrix for evaluating outfits and the web server for accessing images. The latter is part of the reason he came back to work on the application with me near the end. Being full-stack allowed him to have an understanding of how both sides of the project worked. He was able to implement the features in the application that interface with the web server in only a few days.

James and I had a solid working relationship as the two group members working on the application side of the project. With his understanding of the application as a whole he often identified

forgotten points or communication mismatches in our project, and brought them up with either me in my many roles or the group as a whole to resolve together.

Vitor Mendonca

Vitor took on the role of neural net developer, or neural net expert. From the beginning, Vitor took charge of learning all about the DeepFashion dataset, how the neural net works and how to implement one. Equipped with this knowledge, Vitor continued his expertise with a developer role, implementing the neural net mostly, if not completely, on his own.

Vitor and I did not work together too frequently, but when we did I was usually doing tasks under the BA role, such as helping him find a solution for the skewed data. We also discussed project implementation earlier on using the DeepFashion database as a guide. Database implementation may have been the only technical topic we discussed since the database connects all parts of the project together.

Version Control Narrative

Our system is composed of three main components: the mobile application, similarity algorithm and the neural network. We created a repository for each section and worked mainly individually on each of them. As it turns out, phases of work were weekly, similar to a sprint, starting on the week of February 12. All the commits can be found in the Weekly Git Commits Log folder.

February 12

The application repository was created first as design decisions about the application stack were made earlier on. Our first priority was to set up the Ionic Framework and have a default app (provided by Ionic) working.

February 19

Once the simple app was working, progress began on app components shown in the mockups. By the end of the sprint the sign in, registration and main stylist view were present, although design and proper functionality were missing. Only the elements of the view were present.

February 26

Throughout this sprint, the rest of the views were placed and the design of them began. By the end of the sprint sign-in, registration and the stylist side of the app were designed. Additionally, camera work came into play so users can take and save photos in the app.

March 5

In this week more of the app design was implemented, including more of the client profiles in the stylist side, the welcome screen, the client request list and the upload images view. An early implementation of the feature of removing images was completed in the stylist's client profiles, but was promptly moved to the upload images view, a more appropriate place for it. Camera work started in the previous sprint was completed in this stage.

The neural net repository was started by implementing the Convolutional Neural Network tutorial from TensorFlow. The tutorial was adapted to categorize the DeepFashion "type" attribute (top, bottom, full body). Later in the sprint the VGG-16 architecture from the Simonyan & Zisserman paper was implemented in TensorFlow.

March 12
This sprint was about bringing an additional layer of functionality to the app with the addition of Firebase. As designing the views was completed in the previous sprint, this sprint began the implementation of the functionality for each of the views.

In the neural net, there was some trouble batching the images in TensorFlow. This was fixed by scaling the images to 24x24 using OpenCV.

The repository for the similarity algorithm (recommender) was started in this week as well.

March 19
For the app, this sprint was about bringing empty states to the application when the data from Firebase was empty.

In the recommender, the initial similarity function was made and tested. The function worked mainly on attributes. During evaluation of the function's results, we found out about the inaccuracy of the dataset. We chose to look at the 50 categories over the attributes, which was implemented before the end of the sprint.

In the neural net, work was done to help automate the process of generating new samples and to accept custom training files. Additionally work began to move towards the 50 categories and implementing the error function.

March 26
In this sprint we began the last 2 main tasks in the application, filtering the data and completing button functionality. At this point many of the buttons in the app still were not implemented. Two issues were created to help plan and track these two features. By the end of the sprint all the buttons except uploading and downloading of images from the server, and all the filtering was completed.

In the recommender, GET and POST requests were being tested and Firebase connectivity was added.

In the neural net we faced issues with skewed data. We chose to trim the 50 categories into 23 categories. After a script was made to make samples that were equally spaced, all further design of the architecture was stopped in order to allow for training time.

April 2
In this sprint the final button feature (uploading and downloading images) was completed. Additionally, testing and debugging began on an Android device. Most of these bugs were design bugs that Ionic

handles differently on Android. While working on these design bugs, other design elements of the app were touched up. Additionally, a new flow for uploading pictures from the stylist was created so that the stylist can tag the image.

Work in the neural net repository was stopped in the previous sprint so there was no development here.

Along with the button feature of the app, the entire data pipeline of the project was straightened up. From uploading and downloading of images, to those images being sent to the neural network, to those images being placed in MongoDB and in the HTTP Server. Most of this was done in the recommender. Most of the commits here was submitted by Vitor, but was actually James just using Vitor's account.

April 9
There was only minor development this sprint.

Work/Defect Tracking

A Kaban Chart provided by Github was used as our work tracker. Each time some piece of work had to be completed, it would go in either the unassigned Todo column or one of the assigned named columns. When that work was completed, it would be moved to the Done column.

The Weekly Defect Tracker Log folder contains the creation and movements of all work logged into the Kaban which represents all the work tracked for this project. Unfortunately, we only began tracking work as of the March 19th sprint, so the previous 5 weeks of development is not present.

Some of the commits in the commit history mention "bugs" and "fixes" but looking at the commits these aren't really problems with the functional requirements rather design and user experience improvements. My role as designer is to see these as bugs and to fix them, however as a developer and for the purposes of this section, they are not applicable.

These table contains additional tracking:

Arjun's Bugs

	Description	**Commit Closed**	**Severity**	**Date**
1	An error in show/hide logic of the stylist upload tagging hid important content that the user needed to interact with.	e725c94052d5124624b8f84f1ed0a0967231956f	High	04/10
2	On signin, the user was not notified that their authentication parameters did not work, this is fixed with error messaging.	678e7a2f88183870bdba62074926f0c3c0aa2c1d	Low	03/22
3	On signing, Firebase did not distinguish between stylists and clients and would thus sign in stylists into client-side and clients into stylist-side.	001765a822f4b1ec89553e3d510f6820f75f3b1d	High	03/18
4	A new data schema in the backend was made, but the	f75eb64ca47901af9	High	03/14

	front end was never changed to reflect these.	83cdf91a7fbaf03fa1e9c3a		
4	View was broken/hidden due to due a previous restructuring of the HTML content. This was redone and restyled.	9d2f7634b009c464f15c06ffdc1603e51ae4bb39	Medium	03/6

James' Bugs

	Description	**Commit Closed**	**Severity**	**Date**
1	Multiple HTMLs pages and multiple javascript. Don't know how each page communicate to the next without a server section	63f662f2701f377e2d9a765de8833d53e7209c62	Medium	02/27
2	View switch lags, irresponsive. Sometimes it updates, sometimes it don't	771be7158fe8ad65c7ec453945a3d085f81345f2	High	03/07
3	UI router is not directing views/html files correctly	df29d3e0ddac0f5ee99c99e5b2d8630d12d6352f	High	03/07
4	Use MongoDB to host user sign-in and registration on Cybera and try to connect the app to it. Use mongoose to do the api calls. However, the app often crash or never execute any of the mongoose calls.	4cd68b64d42e1d4f8f343a3dc3f99a29c6556717	High	03/12
5	Python standard server package unable to handle post requests	b21e78a04ae57add4f9c348cfd23e76dce7be4f9	Medium	04/01
6	Ionic app cannot display images using src="http://server/img.jpg". Every icon/images on ionic app require a GET request, which is very slow and tedious and require a lot of changes on the ionic app	d1e5c7fdbcb0d54d95d3aed67aff1fa89b62d93b	High	04/04
7	Last minutes issue, the neural network cannot categorized multiple pieces on a single outfit	670a64bde269ae545cfa22ccc2d39353d91b84cb	Medium	04/08

Vitor's Bugs

	Description	**Commit Closed**	**Severity**	**Date**

1	OOM when running VGG-16 (Trying to run the neural net causes an Out Of Memory error at run time - the NN exhausts the ~11GB of internal memory in the GPU. This can probably be fixed by batching - need to figure out how to do it.)	No commit	High	04/10
2	Tensorflow spawns thousands of threads (When running the training script, the Tensorflow engine spawns too many threads. If the sample is large enough, it can cause the program to stop because of exhausted system resources)	c3a150c521c1199712cc5fb4b06c098c9017c8e3	High	03/11
3	CUDA initialization error (This error suddenly started showing up. I try to run the top_bottom_trainer and the result is either [large error message omitted] or [large error message omitted]. This persists even if I revert my changes or reboot the instance.)	No commit	High	03/18
4	Loss tensor created by sigmoid cross-entropy is a vector (We need a single number to input it into the Gradient Descent API)	ca6a30f65196405ca16e2fddcd52f92ebd1eb223	Medium	03/18
5	Sample file is sorted (After generating a sample, the resulting sample file is sorted. If the sample file is subsequently split into smaller files to train in separate section, images with same tag will be clustered together, skewing the training)	eb3e0beac9e53482505fbcf6295906f181de6e9b	Medium	03/18
6	Validation samples are too big (10% of a sample can be excessively large (and wasteful) if the sample is large. It should have a limit of 1500 or so)	15a92fab4c5e02e9bb7ca8a24964d708451129e6	Low	03/18
7	The mean cross-entropy loss function in attribute tagging does not converge (Even after 3000+ steps, the loss keeps alternating between 0.050 and 0.070. We need a new loss function) This bug was a no-fix, because we moved away from using attribute tagging.	No commit	High	03/25