ProtoAI: Model-Informed Prototyping for AI-Powered Interfaces

HARIHARAN SUBRAMONYAM, University of Michigan, USA
COLLEEN SEIFERT, University of Michigan, USA
EYTAN ADAR, University of Michigan, USA

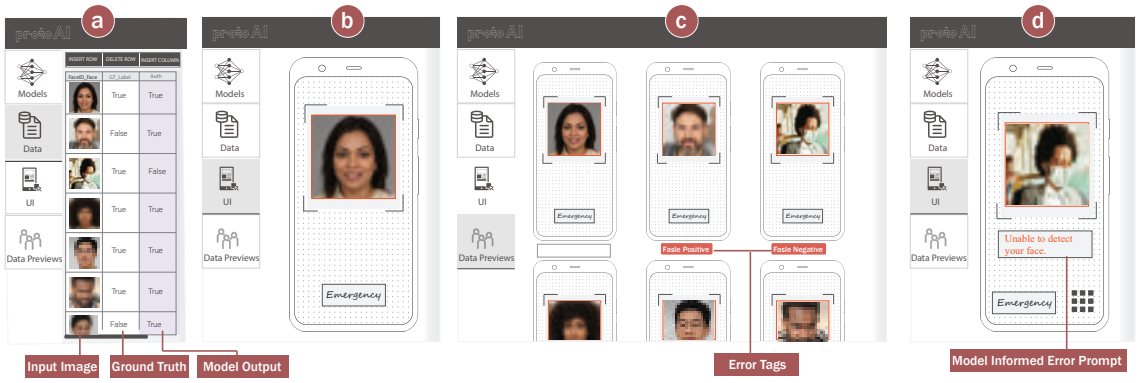

Fig. 1. Prototyping a Face-ID Phone Unlock user interface. The designer (a) provides a set of portrait photos and runs the Face-ID model, (b) prototypes the interface using a design-by-instance approach; (c) ProtoAI generates instances of the interface for all of the photos and flags false positives and false negatives. The designer (d) updates the design to include a numeric keypad and shows error message based on AI model output.

When prototyping AI experiences (AIX), interface designers seek useful and usable ways to support end-user tasks through AI capabilities. However, AI poses challenges to design due to its dynamic behavior in response to training data, end-user data, and feedback. Designers must consider AI's uncertainties and offer adaptations such as explainability, error recovery, and automation vs. human task control. Unfortunately, current prototyping tools assume a black-box view of AI, forcing designers to work with separate tools to explore machine learning models, understand model performance, and align interface choices with model behavior. This introduces friction to rapid and iterative prototyping. We propose *Model-Informed Prototyping* (MIP), a workflow for AIX design that combines model exploration with UI prototyping tasks. Our system, *ProtoAI*, allows designers to directly incorporate model outputs into interface designs, evaluate design choices across different inputs, and iteratively revise designs by analyzing model breakdowns. We demonstrate how ProtoAI can readily operationalize human-AI design guidelines. Our user study finds that designers can effectively engage in MIP to create and evaluate AI-powered interfaces during AIX design.

CCS Concepts: • **Human-centered computing → Systems and tools for interaction design**; **Interface design prototyping**; • **Computing methodologies → Machine learning**.

Additional Key Words and Phrases: AI-Powered Interfaces, Human-Centered AI, Design-by-Instance

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2021 Association for Computing Machinery.
Manuscript submitted to ACM

ACM Reference Format:
Hariharan Subramonyam, Colleen Seifert, and Eytan Adar. 2021. ProtoAI: Model-Informed Prototyping for AI-Powered Interfaces. In *26th International Conference on Intelligent User Interfaces (IUI '21), April 14–17, 2021, College Station, TX, USA.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3397481.3450640

1 INTRODUCTION

When prototyping potential designs for user interfaces (UI), designers work to transform end-user needs into interface specifications [67]. By taking a *top-down* approach, designers: (1) express user requirements as task-flows; (2) map task items into graphical user interface (GUI) objects; and (3) assess different task-to-GUI mappings against end-user needs to finalize the design [45, 67]. For instance, to design a phone 'unlock' user experience (UX), the designer may consider interface alternatives—such as an alpha-numeric password, a numeric passcode, or pattern-based unlocking—to allow end-users to input identifying information for access. By assessing those alternatives against user needs (e.g., fast to unlock, secure, low cognitive effort to remember), the designer will finalize the UI design. However, when prototyping AI-powered applications, such a top-down approach is impractical [71].

AI-powered applications bring additional challenges to UI prototyping. AI features introduce dynamic behavior due to the scope of training data, system use over time, and variations in input data individual users contribute and the potential to learn from outcomes. Thus, designers must identify the *interactions* between user task-flows and AI capabilities [14, 16, 35] in order to design the user interface for AI experiences (AIX). By exploring AI's capabilities and limitations through prototyping, they need to design interface adaptations such as explanations for AI outputs, seamless handling of AI failures, and collecting user feedback to improve the AI [4]. In the process, AIX designers also need to assess interface choices against diverse users and contexts of use.

Unfortunately, current UI prototyping tools lack support for designing AI-powered interfaces [68]. By assuming a 'black-box' view of AI, tools make it challenging for designers to access necessary AI attributes during the design process [62]. Prototyping tools also lack support for iterative testing of AI features through a "fail fast, fail often [69]" approach. For a *AI-powered* phone access using face identification (ID), current tools can at best show where to display the camera field of view on the interface and design static error messages. However, without exploring the AI's behavior first-hand, the designer may not know what inputs the AI needs (e.g., head frontal-view). They may fail to understand how accurately the AI can perform, when it might fail, and how to prompt users experiencing failure (e.g., by asking them to move closer to the camera). To prototype AI features, designers currently need to work with multiple tools to explore AI behavior (e.g., [51]), probe its capabilities and limitations [24], and evaluate their design with diverse user inputs (e.g., skin-tone, lighting conditions, camera angle, facial features such as beard, glasses, or a mask) [11]. This introduces friction to the rapid prototyping process [5, 56]. Thus, the motivating question for our work is: *How might prototyping tools allow designers to directly incorporate target AI features during rapid and iterative prototyping?*

Through an analysis of current human-AI (HAI) design guidelines from academic and industry sources [4, 23, 37], we identified a set of needs and design considerations for AI prototyping tools. To maximize end-user success with AI features, designers need to optimize UI design through *vertical* end-to-end prototyping [5]. They also need to identify different kinds of interface *breakdowns* such as mismatch with end-user expectations, low utility (high cost) from using AI, and data specific failures and offer repairs to recover the user experience. Collectively these tasks require that designers can simulate their interface designs with different data and model outputs. To accomplish this, we propose *Model-Informed Prototyping* (MIP), a workflow that combines model exploration and interface design tasks.

In our system implementing MIP, *ProtoAI*, designers can directly run target machine learning models by providing input data and then incorporate the model's outputs in their UI prototypes. Instead of placeholder content, ProtoAI's *design-by-instance* approach allows designers to experience the AI's behavior first-hand as they are designing. Further, ProtoAI automatically generates data previews of the UI for differing input data, allowing designers to evaluate designs for breakdowns across diverse scenarios and contexts of use. This enables them to decide how best to integrate AI features into end-user's tasks and offer necessary adaptations for AI's uncertainties. As shown in Figure 1a, to design a Face-ID phone unlock AIX, the designer can begin with a diverse set of registered and new faces along with ground truth data as inputs to the *Face Identification* model. After running the model, the designer can prototype the Face-ID user interface using one of the input faces and corresponding model outputs (Figure 1b). In the data previews tab, ProtoAI automatically generates previews of the interface for each input data, allowing the designer to evaluate the designed AIX for diverse inputs (Figure 1c). ProtoAI automatically tags errors such as false positives and false negatives based on ground truth data. By analyzing errors, the designer can revise the interface design by providing alternative login options and displaying data specific prompts to recover from errors (Figure 1d).

MIP streamlines model exploration and UX design tasks during the prototyping process for AI-powered interfaces. By extending the familiar design paradigm of current prototyping tools, ProtoAI allows designers to operationalize HAI design guidelines within their created designs. Based on feedback from designers, ProtoAI lowers the barrier to data-driven design required in prototyping AI features. Our key contributions include: (1) *Model-Informed Prototyping* – a new workflow for prototyping AI-Powered applications, (2) ProtoAI, a tool that implements MIP for GUI, (3) results demonstrating how our approach can support different types of AI breakdowns and repair.

2 RELATED WORK

The user interface design process consists of a series of transformations between end-user task requirements and the user interface syntax [67]. Standard UI prototyping tools such as Wireframe.cc [66], Figma [20], and Adobe XD [1] allow designers to work at the user interface level alone through *horizontal* prototyping [5]. However, when designing AI-powered applications, both the end-user task requirements and the underlying AI components needs to be mapped onto the user interface syntax [12, 14, 23]. This requires a form of *vertical* prototyping in which designers can access specifications about the underlying AI implementation and map them to AI-powered interfaces [5, 64]. In ProtoAI, our goal is to address this need by designing a vertical prototyping tool for AI-powered interfaces. A recommended workflow for UI prototyping consists of three phases: design, test, and analysis. A number of UI prototyping tools (including our own) follow this model [29, 38, 42]. Here, we describe requirements and techniques from prior literature for each phase as applied to AI-powered interfaces.

2.1 Design

Numerous guidelines exist to design AIX by considering the intersection of human-centered needs and AI capabilities [4, 21, 23, 31, 36, 37, 63]. However, to *operationalize* these guidelines, designers need access to the AI model in order to map its characteristics to the UI syntax [16] (see Section 3). For instance, in mixed-initiative design, AI systems automatically act on end-users' goals (when clear) and use interface 'dialog' to resolve any uncertainty [36]. However, the specific dialog in the UI depends on the underlying AI and input data-context. In this regard, prior work has looked at using data as a material for AI design [32, 40]. Just as engineers prototype ML models, designers can begin with 'minimum-viable-data' and iteratively incorporate additional data for diverse users and contexts [46, 62]. This allows prototyping of AI interfaces from the inside-out: from the data model to UI [2]. Mixed-fidelity prototypes [49] could

allow designers to incorporate high-fidelity data elements in early-stage prototypes to represent ML's dependency on data [17]. In ProtoAI we take a similar approach and allow designers to incorporate input data and ML model outputs into UI prototypes (e.g., designing password meters by mapping scores from neural networks and heuristics to a visual bar [61]). Further, given most designers' limited expertise with AI [70], prototyping tools should make AI features more accessible, immediate (support rapid iterative feedback, reflection-in-action, and reflection-on-action), and generative (allow test, probe, and exploration iterations) for UI designers [13, 27, 41].

2.2 Test

AIX designers need to map AI-to-interface features, identify gulfs of execution and evaluation, and assess visual aesthetics for AI features. Further, they should evaluate whether their design is robust to AI's unpredictability [35]: How does the AI-infused interface react to a diverse set of data and contexts of use [12, 62]? Building on existing UX practice, designers may consider approaches such as constructing personas with varying quantitative data [53]. Wizard of Oz (WoZ) testing is also effective for evaluating early-stage prototypes [8, 18, 48], and a number of data-dependent systems implement digitally scaffolded 'wizards' for testing prototypes during design [15, 29, 38, 42]. For instance, Suede implements electronically supported WoZ testing techniques that generate chat messages using test data [38]. In Topiary [42], designers create a map that models people's location, which the Wizard uses to update locations during testing. In ProtoAI we automatically generate interface alternatives by invoking built-in models with input data provided by designers. This lets designers experience the UI's design first hand [9]. In addition, conventional interface design methods include indicating how the UI should behave through demonstration by examples (e.g., [52]). Inspired by this approach, in ProtoAI, we allow designers to configure desired behavior (ground truth) by providing model output data for comparison (i.e., designer as wizard [8]).

2.3 Analyze

To analyze performance at the AI model level, engineers use summary statistics such as accuracy, precision, and recall. Tools exist for engineers to analyze the overall performance and look at individual data points to reason about model failures (e.g., [3]). Designers need similar analysis and visualization tools at the interface level that will allow them to identify mismatches in model behavior. For instance, D.tools offers a 'group analysis' mode aggregating data from multiple user sessions into one view [29]. The What-if tool [24] allows designers to see the confusion matrix for binary classifiers visually [24]. Designers should also be able to incorporate subjective metrics at the intersection of model performance and UX (e.g., subjective perception of errors [39]). In ProtoAI we support subjective analysis through designer generated tags and visual summaries. During iterative prototyping, the goal is to identify breakdowns in design and offer fixes [7, 22, 26, 54, 65]. For instance, through iterative UI experimentation, Quick Access identified UI needs to offer proactive recommendations [60]. The DECOR system characterizes multi-device responsive UIs as a design repair problem and offers techniques for efficient repairs [57]. ProtoAI's instantiation of UI for different data points allows designers to analyze AI-feature breakdowns without performing mental simulations of differing data contexts. Moreover, the generated previews provide the necessary context to make effective repairs [30].

3 DESIGN CONSIDERATIONS

A primary objective when prototyping AIX is to maximize end-users' success. In this regard, both academic and industry sources have put forth design guidelines about good AIX design [4, 23, 37]. With ProtoAI, we want to make it easier for designers to operationalize these guidelines in their interface designs. We collected a total of 284 Human-AI design

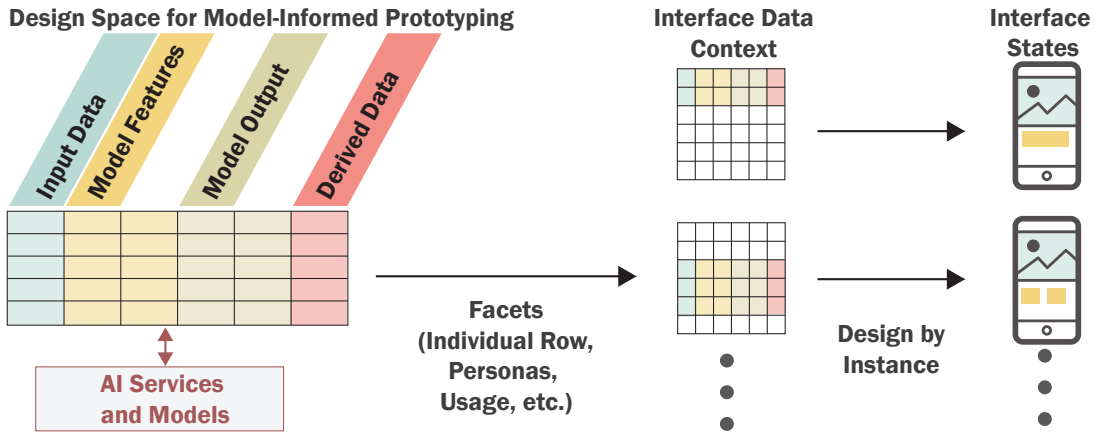

Fig. 2. Design space and workflow for Model Informed Prototyping.

guidelines. We conducted inductive *in-vivo* coding to synthesize the main objectives and tasks for designers and the corresponding AI components necessary to accomplish those tasks. We find that the guidelines offer best-practice recommendations to map AI features into UI design patterns (and end-user tasks). This includes making decisions about automation, AI assistance, and human-effort by aligning AI capabilities and end-user needs. More importantly, the guidelines prescribe design 'fixes' to lower end-user impact from AI-breakdowns such as (1) *end-user context breakdown*: AI performs poorly for some user-data and in some usage contexts; (2) *expectation breakdown*: AI behavior and outputs do not align with end-user mental models; and (3) *task-utility breakdown*: higher cost of using AI due to its failure to understand end-user goals. To address these breakdowns, designers need access to the underlying AI model, features, and output data for diverse end-user inputs. On this basis, we derived a set of design considerations for AIX prototyping tools (i.e., model-informed prototyping).

D1: Prototyping tools should allow designers to invoke ML models by specifying input data directly. When prototyping AI features, designers need to choose whether to automate the task entirely, ways to augment human effort with AI, and whether the AI should be proactive or reactive (acting only upon human invocation), etc. [37]. The objective is to minimize interference with end-user tasks while maximizing utility [36]. To make these choices, designers need to understand how the AI performs for different input data, what output it returns, and under what conditions it might fail. This will allow designers to incorporate AI features into end-user tasks appropriately. Further, designers need access to AI model features to offer the rationale behind specific outputs for specific users. For instance, they may want to present confidence in the model's output or show "why" messages to end-users to design for transparency and trust. To realize these design objectives, designers need to evaluate model performance for potential end-user inputs during their prototyping process.

D2: Prototyping tools should allow designers to incorporate AI outputs into interface design. When designing AI-powered interfaces, guidelines recommend that the AI outputs and UI presentation be aligned to avoid cognitive distortion. Further, the AI-generated content should be visually different to allow end-users to adjust their expectations about AI features (and, in turn, diminish frustration). In mixed-initiative design, designers need to find the right presentation based on confidence thresholds (e.g., showing only the high accuracy item, ranking items as a list, etc.). Designing for these guidelines could benefit from instance-based prototyping by directly incorporating model

outputs into interface design. This will give designers a more accurate representation than placeholder elements when making design choices (e.g., presentation layout, conditional logic for UI, error presentation, guided-recovery from failure, feedback controls, etc.).

D3: Prototyping tools should allow designers to shape model APIs according to end-user needs. Based on decisions about AI feature integration into interface design, designers may need to revisit the model inputs and outputs (i.e., the API). Design guidelines recommend that AI model APIs be designed based on principles of information architecture for interface design. For instance, designers may need to split complex outputs and explanations into multiple parts and present them one at a time. When presenting statistical or numeric outputs, the designer needs to consider factors such as precision and rounding. In cases when numeric values are not appropriate, designers should determine appropriate mappings to categorical variables. This is particularly useful when presenting recommendations along with explanations [23]. Prototyping tools should allow designers to flexibly transform model output and feature values into end-user-friendly formats.

D4: Prototyping tools should allow designers to evaluate design choices across diverse users and usage contexts. With conventional applications, design typically *converges* to a set of standard features across all users. However, with AI models, we can personalize the end-user experience to highly specific contexts. With this intent, HAI guidelines recommend applications should be designed to work across a diverse set of users, use cases, and contexts of use. For example, "while all errors are equal to an ML system, not all errors are equal to all people. [23]" To operationalize these recommendations, designers need to evaluate their interface choices across diverse data. Prototyping tools should allow such evaluation to test and analyze the impact of unwanted model behavior that could negatively impact users. Tools should also support evaluating how AIX could evolve over time and how the interface should adapt accordingly.

D5: Prototyping tools should allow flexibility for designers to incorporate model-related data rapidly and iteratively. Based on our analysis of the design guidelines, we formulate a design space for Model-Informed Prototyping. As shown in Figure 2, MIP's design space is comprised of (1) end-user input data to ML models, (2) model features, (3) model outputs, and (4) the designers' derived data from model outputs. Further, this space can be projected (faceted) into interface data contexts and can include individual input data points, all data for a given end-user (persona), data-contexts that indicate temporality etc. Third, a data context can be bound to an interface state. Designers can evaluate the design for diverse users and contexts by generating interface previews for different data contexts. When prototyping for AI features, tools should allow designers to navigate across this design space flexibly. Designers should be able to switch between model simulation, design, test, analysis, and revision and repair.

4 MODEL-INFORMED PROTOTYPING

Based on design considerations, we implemented ProtoAI to prototype AI-powered interfaces for AIX design. ProtoAI consists of four main views: (1) an AI models and services view (these can be implemented AI services or models, or Wizard of Oz 'stubs'), (2) a data view to import diverse input data for model simulation, (3) a UI 'designer' view to visually construct the interface prototype, and (4) a data previews view to simulate the interface design across different input data contexts. To better understand how a designer might use ProtoAI to engage in MIP, let us follow Divya, an AIX designer who is prototyping a Face-ID-based phone unlock experience.

4.1 Set-Up

Divya opens the ProtoAI application in the web browser. The Models tab is open by default and shows all of the AI services and models that are available in the system (Figure 3a). Divya's company has already assigned an engineering

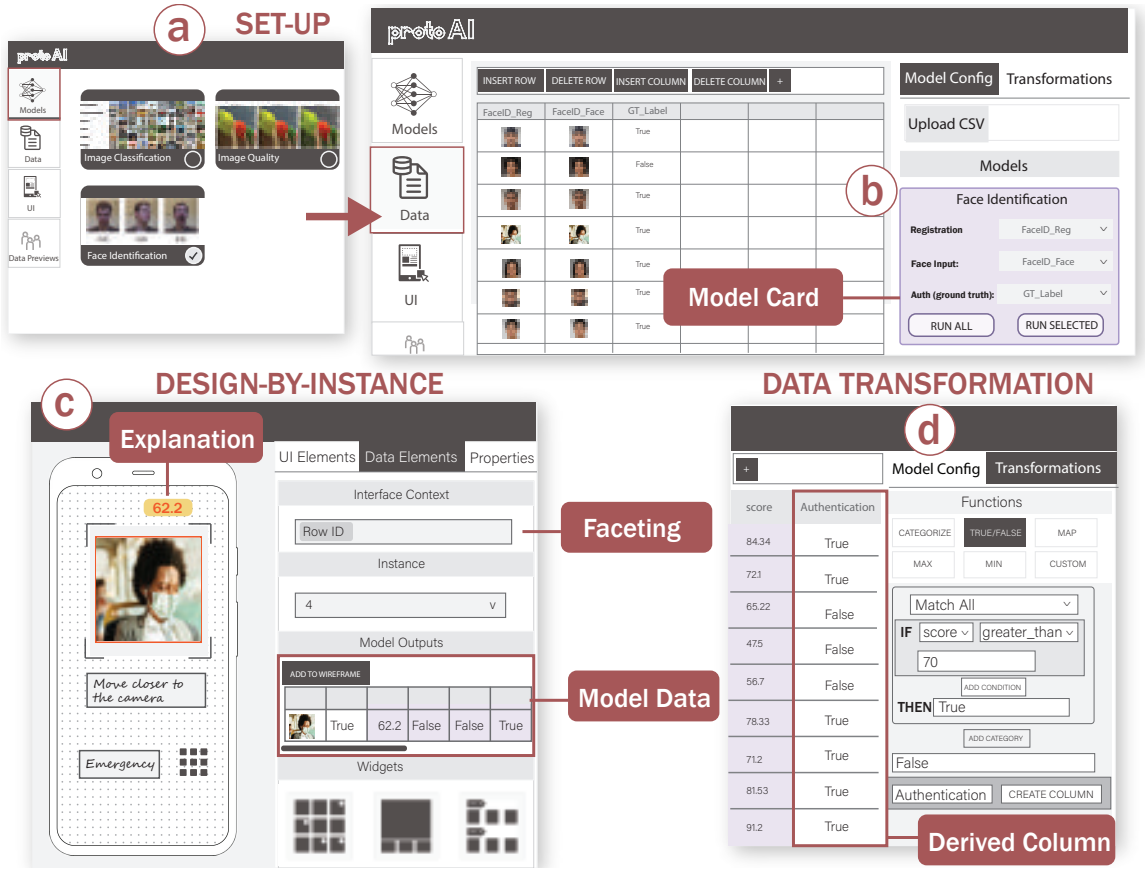

Fig. 3. ProtoAI's user interface and features for MIP. To set-up, the designer (a) selects the Face-ID model, and (b) configures it using the model card. In the User Interface tab, the designer (c) incorporates model inputs and outputs in the wireframe; and (d) transforms Face match score into Boolean column in the Data tab.

team to the project, and they have been working on an initial version of the Face-ID model. Divya selects the company's Face-ID model and navigates to the Data tab. The Data tab will allow her to import input data for different personas and scenarios of use. As shown in Figure 3b, the Data tab consists of a main editable spreadsheet view and sidebar view for model configuration. The spreadsheet can consist of *input data columns*, *feature/parameter columns*, *AI output columns*, and *derived (calculated) columns*. Column types are made distinct through color-coding. The sidebar view shows a model card [50] for each model selected. From the Face-ID model card, Divya sees that the model requires images (both for training/registration) and optional ground truth labels.

Based on her user research, Divya has curated a set of personas and portrait photos for each persona taken across different usage context (e.g., low light condition, crowded subway, person with a beard, facial hair, different skin tones, etc.). Divya can manually input data into the spreadsheet or import it from external sources (e.g., a CSV file). To simulate the model with this data, Divya maps the column headers on the spreadsheet with the model card inputs by selecting from a dropdown list of all columns. These images become the input data columns. Once configured, Divya runs the Face-ID model for the imported data (aligned with design consideration D1). ProtoAI extends the spreadsheet

and appends additional columns with model outputs. The model output columns are color-coded to match the model configuration card. In an alternate scenario, in the absence of a pre-existing model, Divya can use the spreadsheet view to "draft" desired model behavior and outputs for different input data and share those specifications with her engineering team. The Face-ID model that her engineers have created return additional details: a percentage match score (calculated based on face distances in the face embedding space), an explainable heatmap rendering of the input image [55], and a set of Boolean flags for model features (e.g., whether a face was detected, eyes were closed, etc.). Using this simulated output, Divya can proceed to design the user interface for Face-ID based unlocking.

4.2 User Interface Design

Divya selects the User Interface tab, which consists of a design canvas and a sidebar for interface elements. The design canvas starts with a default phone template, but Divya can select others if needed (e.g., desktop or tablet). The sidebar consists of three panels, including the *UI Elements* panel which had a set of standard interface elements, the *Data Elements* panel which hosts input and model output data and a collection of widgets for MIP, and a *Properties* panel to set element-specific properties. To design the phone unlock experience, Divya wants to show the camera view in full screen, along with a button at the bottom for emergency calls and an icon on top to indicate the phone is scanning for a face. Divya first adds the emergency button by selecting the button element from the UI elements tab. She also adds a placeholder image on top of the screen to represent scanning status.

Next, to engage in design-by-instance prototyping, Divya opens the data elements panel (Figure 3c). This panel consists of a faceting control to set the wireframe's *data context* and a table showing the faceted data itself (a subset of the main spreadsheet view). The data context is the scope of end-user data that will be bound to the interface at runtime. The faceting feature is flexible and can set the data context to a single row or a set of rows nested and grouped by column names. For example, in a different scenario, Divya can set the data context to all images a persona has taken (e.g., for a photo album interface). Because the Face-ID UX shows the camera feed from the front-facing camera (i.e., a person's face), Divya sets the data context to a single row.

From the faceted table, Divya selects the cell value with the persona's face image and clicks on the 'Add to Wireframe' button. ProtoAI adds the image of the person's face onto the template, and Divya can adjust it to fit her design. Divya also adds the percentage match score value from the Face-ID model's output to the interface (from design consideration D2). While not intended for the final deliverable, Divya can use it to test and debug the interface design. To indicate this to ProtoAI, Divya toggles the 'set as explanation' flag in the properties tab for the score element. This will allow her to selectively show the explanation overviews later in the previews tab. For other complex layout needs, Divya can select entire columns, or brush select the desired data from the data table and add them to the interface as a widget. Based on AIX interface design patterns, ProtoAI implements an initial set of widgets for binding Boolean values to images, categorizing items by tags, and showing ranked order of items. Each widget has a predefined layout and can be bound to selected data along with explanation overlays for designers. The widget library can be extended in the future to support additional layout design needs.

4.3 Design Evaluation

At this point, Divya has an initial wireframe of the phone unlock interface designed using the portrait image from a single persona. She selects the Data Previews tab to evaluate her current design against different personas and their photos. ProtoAI automatically instantiates the screen interface based on the data context and using all data imported in the data tab (D4). As shown in Figure 1c, the Data Previews tab consists of a scrolling grid view of the UI rendered for

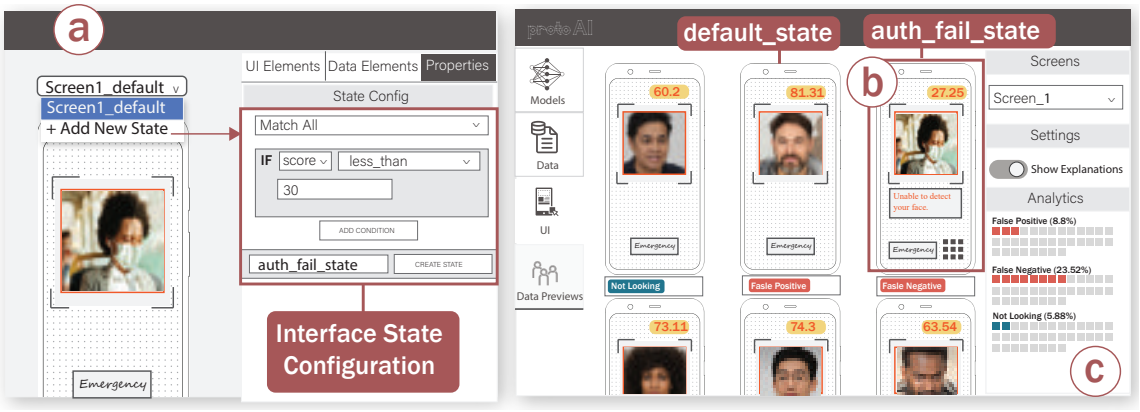

Fig. 4. To design scenarios with Face-ID authentication failure, the designer (a) creates a new interface state conditioned on the match score below 30. In the Data Previews tab, ProtoAI generates appropriate interface states based on scores, and (c) provides analytics for number of instances with errors.

different users and their portrait photo variations. The preview view allows Divya to rapidly evaluate her design as it is being created and conduct design checks. In a different scenario, to evaluate model functionality over time with model learning, Divya can configure data for different personas by providing different amounts of input data. This will allow Divya to visually see how the AI-powered interface responds after differing degrees of use. Divya can also check her design for different data sizes from model output (e.g., recommendations), ranging from no recommendations, a few recommendations, to tens of recommendations. Third, Divya can also evaluate the design for localization by providing inputs in different languages. Fourth, suppose the model's parameters require tweaking (such as the number of clusters). In that case, Divya can configure the data with different cluster sizes and compare the results in the data previews view.

4.4 Analysis, Revision, and Repair

ProtoAI's 'evaluation through previews' is intended to support the designer in analyzing design breakdowns in differing real-world contexts. Using the Data Previews view, Divya can iteratively revise the design to make it robust for a wider variety of users and contexts. ProtoAI offers a number of analysis features to support this iterative MIP workflow. Because Divya specified ground truth data for each of the photos, ProtoAI automatically compares the ground truth (Face-ID match) with model predictions and tags instances of false positives and false negatives. In the sidebar, ProtoAI provides a summary of each tag indicating the number of instances with that tag. Divya can see that in 16% of data, the model predicted an identity mismatch when the image was, in fact, the persona (i.e., false negatives). By checking the 'show explanations' flag in the sidebar, Divya can see the match score element she added in the UI tab. Similarly, Divya can also overlay other model factors and outcome values such as identified facial features or saliency maps to help her understand what the model computed from the image. Divya can also add her own tags to indicate domain-specific types of breakdowns or repairs. In this example, Divya sees that the model fails when the person is farther away or when their eyes are closed.

To address this issue, Divya switches back to the Data tab and creates a new calculated Boolean column that is set to 'true' if the face is not detected or when eyes are closed (D3). The Data tab allows for several different types of data *transformations*, including the categorization of numerical values (e.g., high, medium, and low), mapping transformations

of model-assigned labels and values to end-user-friendly labels, calculating the minimum and maximum values, and custom formula functions for excel-like computation by specifying column headers and cells values to include in the computation (see Figure 3d). Through these transformations, Divya can design the model's API. Put another way, she controls the format in which the model output is presented in the user interface.

After creating the Boolean column, Divya returns to the UI tab to address the false-negative instances (D5). In ProtoAI, each screen can be assigned different screen states dependent on model behavior and values. Divya adds a new interface *state* to the unlock screen conditioned on the Boolean column value, which she configures using the properties panel (Figure 4a). In this state, Divya adds a message at the top of the screen prompting the user to move closer to the screen. In addition, Divya adds a numeric password option to address the remaining failure cases. When she returns to the previews view, Divya sees that instances of false negatives have the prompt message she just created. For interfaces with multiple screen states and screen-to-screen flow, ProtoAI offers a summary view showing a navigation diagram indicating how each end-user (based on their data) would engage in the AI-powered task flow. This allows Divya to see the probabilistic nature of the AI's task-flows to ensure that all users meet the desired goals. In this manner, Divya can design the interface using direct outputs from the ML model, evaluate her design against different AI and real-world constraints, and iteratively revise the design and repair the API to prototype the AI user experience (D5).

4.5 Implementation

We implemented ProtoAI as a web-based application using a client-server architecture. The server was written in Python and hosts different ML models. We use the metadata format in RunwayML [51] to specify the inputs and outputs to the model. Through this metadata, we generate the client-side model cards. This allows a number of already available models to be used in ProtoAI. The client is implemented using HTML and JavaScript. We make use of third party libraries for the UI design canvas [19], the spreadsheet interface [34], and formula parsing [10].

5 DESIGN SCENARIOS

To demonstrate the utility of ProtoAI in operationalizing HAI design guidelines, we offer three usage scenarios based on real-world examples.

5.1 Social Media Feed–Automated Image Cropping

ProtoAI supports testing for, detecting, and fixing context breakdowns during design. A recent example of this need is the image auto-cropping feature offered on social media feeds (e.g., Twitter's problem [33] and their response [43]). We imagine a process by which ProtoAI can be used to fix the bias in cropping. The designer begins by collecting various images with different sizes, aspect ratios, salient points, and content semantics. They curate this data based on user research on photos uploaded to social media. The designer can then run the Auto-Cropping model against those images to view the cropped image within the user interface design. In the Data Previews tab, the designer sees that some images are cropped appropriately, but others leave out foreground objects or show only background. To investigate this issue, the designer can overlay the image saliency map (class activation mapping) returned by the model. Back in the preview mode, the designer sees that cropping fails when there are multiple salient points and no salient points. By tagging the images with appropriate labels, the designer sees that around 30% of images in the dataset have multiple salient points, and 5% have no salient points. To suggest a fix, the designer proposes an image widget for users that pans between different salient points in a loop. To resolve images with no salient points, the designer adds interface functionality for *manual* cropping using the AI-generated crop region as a suggestion.

5.2 Movie Recommendations–Changes with Use

Another challenge for designers is knowing how the interface and user experience may change over time as the AI learns from end-user data and feedback. ProtoAI can help simulate design previews over time and use. For example, we imagine a designer using ProtoAI to design a movie recommendation page. The designer can set up the data for different personas (either real-world preference or simulated)[1]. Based on input data, the model returns a set of movie recommendations for each persona and factors explaining why the movie was recommended. The designer then wireframes an initial interface listing all of the movies recommended by the AI. The previews tab shows that for personas with few or no input data about movies already watched, the recommendations do not align with the fictional persona's preferences. By looking at the confidence score for recommendations, the designer creates a new screen state for low confidence recommendations: instead of showing the movies, it asks end-users to select movie genres of interest. For other personas with sufficient input data and suitable recommendations, the designer chooses to present a categorized output by transforming confidence scores into confidence categories. Further by looking at the explanation factors, the designer can incorporate model explanations in text form: "Because you watched [explanation value], we think you might like:" Through data personas with differing inputs, ProtoAI allows designers to simulate model behavior over time of use, and design appropriate interface experiences.

5.3 Chat Assistant–Mixed Initiative Design

A guiding principle for integrating AI capabilities into task workflows is to determine the utility of the AI for end users [36]. If the AI is confident about the end user's goals, it can tend towards automation. If the goal can be resolved with minimal support from users, the AI can engage in a mixed-initiative dialog with end-users. In all other cases, AI should not automate the task. This design profile for HAI applies to a variety of AI-powered application designs; yet for designers, understanding the utility function is challenging. The design-by-instance, and Data Preview features in ProtoAI can help designers achieve the mixed-initiative design. For instance, consider a chat assistant's design that prompts end-users with task actions based on chat messages. When the user comments, "Let's meet at Bob's Burger place," the AI can pull up directions to the location if confidence is high. If there are multiple outlets, the AI can present a list sorted by proximity and ask the end-user to pick a specific location. In other cases, the AI should ask the end-user to manually input the address. In ProtoAI, the designer can curate such chat messages for the ML model. In the Preview tab, they can tag instances with incorrect recommendations and overlay each recommendation's confidence score. Later, they can create different screen-states for mixed-initiative and manual inputs using confidence score thresholds. The preview section allows the designer to experience first-hand the subjective utility for end-users and then offer necessary adaptations to their interface design.

6 PRELIMINARY EVALUATION

To gather feedback on ProtoAI's implementation of MIP, we conducted a preliminary online user study. We aimed to (1) assess whether designers can successfully leverage ProtoAI's features for prototyping AI-powered interfaces, and (2) collect feedback on the model-informed prototyping workflow. We recruited 10 UX designers for the study with expertise in prototyping user interfaces using off-the-shelf tools such as Figma, Sketch, Axure, and Adobe XD. Six participants had prior experience designing AI-powered applications. Each session lasted 75 minutes, and participants

[1] An alternative strategy for the designer would be to duplicate the persona rows and remove data. This simulates earlier versions of the persona with less training data

Fig. 5. Example designs generated by our study participants for AI based photo recommendation for Instagram.

were compensated with $20 for their time. At the start of each session, we provided an overview of MIP. We gave an in-depth walkthrough of ProtoAI features using image classification as an example.

Following the walkthrough, participants engaged in a design activity using ProtoAI. We asked participants to design an AI-powered interface to recommend images (from a set) to upload to Instagram. In the interest of session time, we provided data consisting of four personas with five images for each. Each image included a quality score ranging from 0-10 (higher scores indicating better quality) based on a Neural Image Assessment model [59]. We selected images such that different personas had different score ranges and variations in the differences between scores. For each image, we also generated a class activation heatmap [55] to probe participants on the explainability features of ProtoAI. Participants launched ProtoAI on their web-browser and shared their screen during the session. We asked participants to think-aloud during this phase of the session and recorded them. Once participants completed the task, we conducted a freeform interview to understand how they would use ProtoAI for AIX problems they had worked on in the past, provided feedback on ProtoAI's features and interface, and commented on MIP workflow. At the end of the study, participants filled out a usability questionnaire [44] and the NASA-Task Load Index questionnaire [28].

6.1 Findings

6.1.1 Model-Informed Prototyping with ProtoAI. Across all sessions, designers created an image recommendation UX with one or more screens (Figure 5). They used data previews as they worked and created new screen states based on the generated previews. In five of the sessions, designers directly started the activity using data elements, including persona images and quality scores. For instance, in session 10, the designer crafted an initial layout showing only the image with the highest score. Then, by looking at the data tab, they realized there were some small differences between images with the second and third highest scores. They then created two new derived columns to compute the differences in scores and created new screen states for the best two and three images. Designers also used the 'categorize transformation' function to bucket image scores into high, medium, and low categories. In the remaining five sessions, designers first created placeholder layouts and then imported data from the data elements tab. In session 4, the designer created a set of placeholder screens for different data conditions, including one high-quality image, all low-quality images, and all high-quality images. They then replaced the placeholder with real data and model outputs. All designers made use of the explanation overlays. By looking at the CAM heat maps, they revised their designs in ways we did not anticipate. For instance, in session 1, the designer created a new task-flow for end-users to crop the image based on salient regions indicated by the heatmap and re-compute the quality score. In session 5, the designer suggested addressing tasks with no high-quality images by showing the CAM view to end-users and allowing them to retake the photo. Based on the NASA-TLX questionnaire, participants rated the overall task workload of 50.3 (SD = 12.49). A breakdown of individual

components show that participants ratings were: *mental demand*: 59 (SD = 18.07); physical demand: 24 (SD = 21.53); temporal demand:38 (SD = 20.70); performance:38.5 (SD = 18.86); effort:49 (SD = 12.2); and frustration:37 (SD = 16.36).

6.1.2 Utility of MIP. Designers with prior experience designing AI-powered applications and knowledge of HAI guidelines (n=6) saw value in MIP (and ProtoAI). They all mentioned their current data-driven design workflow either by writing code or analyzing data using spreadsheets. In particular, designers appreciated the data-to-interface pipeline through model simulation, auto-generated data previews, and carrying out data transformations during the design process. In providing feedback about the overall workflow, P4 commented: *"Right now I will have my hypothesis about the data and go back to the engineer and ask them to give me the output, but they say that those data instances will not occur, there is a lack of transparency, and there are layers of gates I need to get through before I can do the next step. This tool makes it easy for me to carry out the entire flow on my own."* When commenting about prototyping for data instances, P5 commented: *"The hardest thing about designing for AI is getting the right data. You can make something look good with fake labels and 'ipsum-lorem,' but using real data to mock things up helps you see where things are broken. I think automatically generating the alternatives using the data is very powerful."* For participants new to AIX design, they compared MIP to their current workflows. They commented they needed scaffolding to understand the AI model and outputs and incorporate data elements in their design.

6.1.3 User Experience. Overall, participants found ProtoAI's interface intuitive and easy to use. They appreciated the flexibility and connectedness of end-user data across different tabs (Data, UI, and Previews). P1 commented that ProtoAI is beneficial at the brainstorming stage, where instead of wireframing on the whiteboard, they can quickly input data and desired model output and test out interface alternatives. P8 commented on the explanation overlays, stating they can add model outputs on the interface and flexibly include it in the final design or flag it as "explanation for the designer." Participants made suggestions for section and tab labels, which we incorporated into the final design (e.g., in the prototype used in the study, 'data previews' tab was labeled 'alternatives'). They also recommended having pop-out windows for the data elements tab to avoid scrolling across each row. Based on the usability questionnaire, on a seven-point scale, participants rated ProtoAI's to be easy to use ($mean$ = 5.88,SD = 0.9), and flexible ($mean$ = 5.63,SD = 0.72). Participants rated their learnability (i.e., can learn it quickly) a mean score of 5.33 (SD = 1.65), and learning without written instructions as 3.22 (SD = 1.39). In future iterations, we can support on-boarding through guided walkthroughs and use cases of design guidelines. Encouraged by the overall feedback, we plan to conduct a comparative evaluation of ProtoAI against commercial prototyping tools and assess the quality of design output using ProtoAI.

7 DISCUSSION AND FUTURE WORK

To design user interfaces for AI-powered applications, designers need access to the underlying AI. Therefore, digital prototyping tools should escape the 'black-box' view of AI by incorporating the AI model's characteristics into the UI prototyping process. In this work, we define a new paradigm for UX design for AI-powered applications, which we call AIX. To accomplish AIX design, we have demonstrated how ProtoAI's implementation of *Model-Informed Prototyping* allows designers to (1) directly incorporate an AI's output into their design, (2) test their design across different input data contexts, and (3) iteratively assess and adapt their interfaces for explainability, failure, and model feedback. Based on our evaluation and participants' feedback, ProtoAI allows designers to prototype AI-powered UI, provide just-in-time model simulation and outputs without AI model engineering, and transform model outputs to meet interface presentation needs. In addition, the data-level representations in ProtoAI correspond to engineering representations of the AI service's API. This affords opportunities for communication, negotiation, and co-design between designers

and engineers. Specifically, future work can investigate how AIX designers can drive AI model parameters based on interface features, negotiate model features and outputs necessary for explainability, and communicate discovered failure instances with engineers for model improvement.

End-user data is a critical aspect of MIP. In this regard, ProtoAI offers flexibility for designers to manually input data from user research and simulated data to explore design their hypotheses about AI behavior. Besides, they can directly import data from other human-centered design processes (e.g., Data-Assisted Affinity Diagramming [58]). However, we do not investigate specific data generation needs during the prototyping process in our current work. When prototyping, designers may need access to diverse data to consider both success and failure cases at the AI and UI levels. We are currently exploring ways to support synthetic data generation needs through expressive queries. For instance, visualization design tools allow designers to generate data with specific statistical and visual properties [25, 47]. AIX designers may also need to work with sensor data or implicit feedback collected by system logs. Future work should look at ways to support these specific data and analysis needs and advanced user-modeling for MIP. Third, ProtoAI has the potential to support Responsible AI needs such as fairness, accessibility, and transparency. AI engineers are asked to evaluate their data and ML models for responsible AI criteria (e.g., AI Fairness 360 [6]), and AIX designers can use tools like ProtoAI's data previews to detect interface failures in responsible AI design.

In ProtoAI, we assume that MIP is useful during early-stage prototyping (i.e., generative wireframing). This allows us to trade-off design complexity for detailed data. Further, while ProtoAI supports evaluation by designers through data previews, certain types of experiential design failures may not be apparent to designers. Future research should look at how MIP can be integrated into later stages of AIX prototyping and usability testing workflows. This includes supporting interactive and click-through prototypes, sharing prototypes with end-users, and logging capabilities. Finally, as pedagogy and practice of AI application design continues to evolve, we envision AIX tools like ProtoAI will enable students and novice designers to develop necessary skills for AIX prototyping. We imagine a library of widgets implementing AIX design patterns and explainable overlays to scaffold designers' learning process.

8 CONCLUSION

While AI capabilities are prevalent in everyday and high-stakes software applications, end-users frequently encounter unpleasant AI experiences. A challenge for designers is that their current design tools mainly assume a 'black-boxed' view of AI. This restricts the designer's ability to anticipate and address breakdowns in AIX. To maximize end-user success with AIX, designers should directly work with underlying AI features during the design process. In this work, we present *Model-Informed Prototyping*, a workflow that interleaves AI exploration and UI prototyping tasks. Our implementation of MIP,ProtoAI, allows designers to directly invoke AI models and services, incorporate model outputs into interface design, and iteratively and rapidly evaluate their design choices across diverse end-users and their data context. We demonstrate how ProtoAI can support designers in operationalizing best practice HAI guidelines. Preliminary feedback from designers highlights ProtoAI's potential to empower designers by providing them just-in-time access to AI features.

9 ACKNOWLEDGMENTS

We thank our reviewers and study participants for their time and feedback. We also thank Daniel Weld and Steven Drucker for their inputs on early designs of ProtoAI.

REFERENCES

[1] Adobe. 2020. Adobe XD. https://www.adobe.com/products/xd.html

[2] Robert Akscyn, Elise Yoder, and Donald McCracken. 1988. The data model is the heart of interface design. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 115–120.

[3] Saleema Amershi, Max Chickering, Steven M Drucker, Bongshin Lee, Patrice Simard, and Jina Suh. 2015. Modeltracker: Redesigning performance analysis tools for machine learning. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 337–346.

[4] Saleema Amershi, Dan Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, Jina Suh, Shamsi Iqbal, Paul N Bennett, Kori Inkpen, et al. 2019. Guidelines for human-AI interaction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, 3.

[5] Michel Beaudouin-Lafon and Wendy E Mackay. 2009. Prototyping tools and techniques. In *Human-Computer Interaction*. CRC Press, 137–160.

[6] Rachel KE Bellamy, Kuntal Dey, Michael Hind, Samuel C Hoffman, Stephanie Houde, Kalapriya Kannan, Pranay Lohia, Jacquelyn Martino, Sameep Mehta, Aleksandra Mojsilovic, et al. 2018. AI Fairness 360: An extensible toolkit for detecting, understanding, and mitigating unwanted algorithmic bias. *arXiv preprint arXiv:1810.01943* (2018).

[7] Paul Beynon-Davies and Steve Holmes. 2002. Design breakdowns, scenarios and rapid application development. *Information and software technology* 44, 10 (2002), 579–592.

[8] Jacob T Browne. 2019. Wizard of Oz Prototyping for Machine Learning Experiences. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–6.

[9] Marion Buchenau and Jane Fulton Suri. 2000. Experience prototyping. In *Proceedings of the 3rd conference on Designing interactive systems: processes, practices, methods, and techniques*. 424–433.

[10] Krzysztof Budnik. 2020. Formula Parser. https://github.com/handsontable/formula-parser

[11] Joy Buolamwini and Timnit Gebru. 2018. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency*. 77–91.

[12] Kim Carmona, Erin Finley, and Meng Li. 2018. The Relationship Between User Experience and Machine Learning. *Available at SSRN 3173932* (2018).

[13] Marcelo Cataldo and James D Herbsleb. 2010. Architecting in software ecosystems: interface translucence as an enabler for scalable collaboration. In *Proceedings of the Fourth European Conference on Software Architecture: Companion Volume*. ACM, 65–72.

[14] M Ceconello, D Spallazzo, and M Sciannamè. 2019. Design and AI: prospects for dialogue. (2019).

[15] Richard C Davis, T Scott Saponas, Michael Shilman, and James A Landay. 2007. SketchWizard: Wizard of Oz prototyping of pen-based user interfaces. In *Proceedings of the 20th annual ACM symposium on User interface software and technology*. 119–128.

[16] Dominik Dellermann, Adrian Calma, Nikolaus Lipusch, Thorsten Weber, Sascha Weigel, and Philipp Ebel. 2019. The future of human-ai collaboration: a taxonomy of design knowledge for hybrid intelligence systems. (2019).

[17] Graham Dove, Kim Halskov, Jodi Forlizzi, and John Zimmerman. 2017. Ux design innovation: Challenges for working with machine learning as a design material. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, 278–288.

[18] Steven Dow, Blair MacIntyre, Jaemin Lee, Christopher Oezbek, Jay David Bolter, and Maribeth Gandy. 2005. Wizard of Oz support throughout an iterative design process. *IEEE Pervasive Computing* 4, 4 (2005), 18–26.

[19] Fabric.js. 2020. Fabric.JS HTML Canvas Library. http://fabricjs.com/

[20] Figma. 2020. Figma. https://www.figma.com/

[21] Yolanda Gil, James Honaker, Shikhar Gupta, Yibo Ma, Vito D'Orazio, Daniel Garijo, Shruti Gadewar, Qifan Yang, and Neda Jahanshad. 2019. Towards human-guided machine learning. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*. 614–624.

[22] Fabien Girardin and Neal Lathia. 2017. When User Experience Designers Partner with Data Scientists. In *2017 AAAI Spring Symposium Series*.

[23] Google. 2019. People + AI Guidebook. https://pair.withgoogle.com/

[24] Google. 2020. Visually probe the behavior of trained machine learning models, with minimal coding. https://pair-code.github.io/what-if-tool//

[25] Robert Grant. 2020. DrawMyData a tool for teaching stats and data science. http://robertgrantstats.co.uk/drawmydata.html

[26] Raymonde Guindon, Herb Krasner, Bill Curtis, et al. 1987. Breakdowns and processes during the early activities of software design by professionals. In *Empirical studies of programmers: Second Workshop*. 65–82.

[27] Lise Amy Hansen. 2011. Full-body movement as material for interaction design. *Digital Creativity* 22, 4 (2011), 247–262.

[28] Sandra G Hart. 2006. NASA-task load index (NASA-TLX); 20 years later. In *Proceedings of the human factors and ergonomics society annual meeting*, Vol. 50. Sage publications Sage CA: Los Angeles, CA, 904–908.

[29] Björn Hartmann, Scott R Klemmer, Michael Bernstein, Leith Abdulla, Brandon Burr, Avi Robinson-Mosher, and Jennifer Gee. 2006. Reflective physical prototyping through integrated design, test, and analysis. In *Proceedings of the 19th annual ACM symposium on User interface software and technology*. 299–308.

[30] Mark Hartswood and Rob Procter. 2000. Design guidelines for dealing with breakdowns and repairs in collaborative work settings. *International Journal of Human-Computer Studies* 53, 1 (2000), 91–120.

[31] Jeffrey Heer. 2019. Agency plus automation: Designing artificial intelligence into interactive systems. *Proceedings of the National Academy of Sciences* 116, 6 (2019), 1844–1850.

[32] Karey Helms, Barry Brown, Magnus Sahlgren, and Airi Lampinen. 2018. Design Methods to Investigate User Experiences of Artificial Intelligence. In *2018 AAAI Spring Symposium Series*.

[33] Alex Hern. 2020. Twitter apologises for 'racist' image-cropping algorithm.

[34] Paul Hodel. 2020. The Javascript Spreadsheet. https://bossanova.uk/jexcel/v4/

[35] Lars Erik Holmquist. 2017. Intelligence on tap: artificial intelligence as a new design material. *interactions* 24, 4 (2017), 28–33.

[36] Eric Horvitz. 1999. Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. 159–166.

[37] Apple Inc. 2019. Designing the UI and User Experience of a Machine Learning App. https://developer.apple.com/design/human-interface-guidelines/machine-learning/overview/introduction/

[38] Scott R Klemmer, Anoop K Sinha, Jack Chen, James A Landay, Nadeem Aboobaker, and Annie Wang. 2000. Suede: a Wizard of Oz prototyping tool for speech user interfaces. In *Proceedings of the 13th annual ACM symposium on User interface software and technology*. 1–10.

[39] Rafal Kocielnik, Saleema Amershi, and Paul N Bennett. 2019. Will You Accept an Imperfect AI?: Exploring Designs for Adjusting End-user Expectations of AI Systems. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, 411.

[40] Peter Kun, Ingrid Mulder, Amalia De Götzen, and Gerd Kortuem. 2019. Creative Data Work in the Design Process. In *Proceedings of the 2019 on Creativity and Cognition*. ACM, 346–358.

[41] Germán Leiva, Nolwenn Maudet, Wendy Mackay, and Michel Beaudouin-Lafon. 2019. Enact: Reducing Designer–Developer Breakdowns When Prototyping Custom Interactions. *ACM Transactions on Computer-Human Interaction (TOCHI)* 26, 3 (2019), 19.

[42] Yang Li, Jason I Hong, and James A Landay. 2004. Topiary: a tool for prototyping location-enhanced applications. In *Proceedings of the 17th annual ACM symposium on User interface software and technology*. 217–226.

[43] Natasha Lomas. 2020. Twitter may let users choose how to crop image previews after bias scrutiny.

[44] Arnold M Lund. 2001. Measuring usability with the use questionnaire12. *Usability interface* 8, 2 (2001), 3–6.

[45] Allan MacLean, Richard M Young, Victoria ME Bellotti, and Thomas P Moran. 1991. Questions, options, and criteria: Elements of design space analysis. *Human–computer interaction* 6, 3-4 (1991), 201–250.

[46] Nirav Malsattar, Tomo Kihara, and Elisa Giaccardi. 2019. Designing and Prototyping from the Perspective of AI in the Wild. In *Proceedings of the 2019 on Designing Interactive Systems Conference*. ACM, 1083–1088.

[47] Miro Mannino and Azza Abouzied. 2019. Is this Real? Generating Synthetic Data that Looks Real. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. 549–561.

[48] David Maulsby, Saul Greenberg, and Richard Mander. 1993. Prototyping an intelligent agent through Wizard of Oz. In *Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems*. 277–284.

[49] Michael McCurdy, Christopher Connors, Guy Pyrzak, Bob Kanefsky, and Alonso Vera. 2006. Breaking the fidelity barrier: an examination of our current characterization of prototypes and an example of a mixed-fidelity success. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*. ACM, 1233–1242.

[50] Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. 2019. Model cards for model reporting. In *Proceedings of the conference on fairness, accountability, and transparency*. 220–229.

[51] Runway ML. 2018. Runway ML: Machine Learning for Creators. https://runwayml.com/

[52] Brad A Myers and William Buxton. 1986. Creating highly-interactive and graphical user interfaces by demonstration. *ACM SIGGRAPH Computer Graphics* 20, 4 (1986), 249–258.

[53] John Pruitt and Jonathan Grudin. 2003. Personas: practice and theory. In *Proceedings of the 2003 conference on Designing for user experiences*. 1–15.

[54] Holger Rhinow, Eva Köppen, and Christoph Meinel. 2012. Prototypes as boundary objects in innovation processes. In *Proceedings of the 2012 International Conference on Design Research Society, Bangkok, Thailand*.

[55] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*. 618–626.

[56] Herbert A. Simon. 1969. The sciences of the artificial (1996; Orig. ed. 1969; 2nd, rev).

[57] Nishant Sinha and Rezwana Karim. 2015. Responsive designs in a snap. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*. 544–554.

[58] Hariharan Subramonyam, Steven M Drucker, and Eytan Adar. 2019. Affinity Lens: Data-Assisted Affinity Diagramming with Augmented Reality. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–13.

[59] Hossein Talebi and Peyman Milanfar. 2018. NIMA: Neural image assessment. *IEEE Transactions on Image Processing* 27, 8 (2018), 3998–4011.

[60] Sandeep Tata, Alexandrin Popescul, Marc Najork, Mike Colagrosso, Julian Gibbons, Alan Green, Alexandre Mah, Michael Smith, Divanshu Garg, Cayden Meyer, et al. 2017. Quick access: building a smart experience for Google drive. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1643–1651.

[61] Blase Ur, Felicia Alfieri, Maung Aung, Lujo Bauer, Nicolas Christin, Jessica Colnago, Lorrie Faith Cranor, Henry Dixon, Pardis Emami Naeini, Hana Habib, et al. 2017. Design and evaluation of a data-driven password meter. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 3775–3786.

[62] Philip van Allen. 2018. Prototyping ways of prototyping AI. *interactions* 25, 6 (2018), 46–51.

[63] Danding Wang, Qian Yang, Ashraf Abdul, and Brian Y Lim. 2019. Designing theory-driven user-centric explainable AI. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–15.

[64] James Wilson and Daniel Rosenberg. 1988. Rapid prototyping for user interface design. In *Handbook of human-computer interaction*. Elsevier, 859–875.

[65] Terry Winograd, Fernando Flores, and Fernando F Flores. 1986. *Understanding computers and cognition: A new foundation for design*. Intellect Books.

[66] Wireframe|CC. 2020. A Design Tool Fine Tuned for Wireframing. https://wireframe.cc/

[67] Larry E Wood. 1997. *User interface design: Bridging the gap from user requirements to design*. CRC Press.

[68] Qian Yang. 2018. Machine Learning as a UX Design Material: How Can We Imagine Beyond Automation, Recommenders, and Reminders?. In *2018 AAAI Spring Symposium Series*.

[69] Qian Yang, Justin Cranshaw, Saleema Amershi, Shamsi T Iqbal, and Jaime Teevan. 2019. Sketching NLP: A Case Study of Exploring the Right Things To Design with Language Intelligence. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, 185.

[70] Qian Yang, Alex Scuito, John Zimmerman, Jodi Forlizzi, and Aaron Steinfeld. 2018. Investigating how experienced UX designers effectively work with machine learning. In *Proceedings of the 2018 Designing Interactive Systems Conference*. ACM, 585–596.

[71] Qian Yang, Aaron Steinfeld, Carolyn Rosé, and John Zimmerman. 2020. Re-examining Whether, Why, and How Human-AI Interaction Is Uniquely Difficult to Design. In *Proceedings of the 2020 chi conference on human factors in computing systems*. 1–13.