Application of Visual Availability Theory and Visual Perception in Software

LiuFei[1,a], JiangYe[1,b], LiuXianJun[1,c*]

[1]CHN ENERGY DIGITAL INTELLIGENCE TECHNOLOGY DEVELOPMENT (BEIJING) CO., LTD，Beijing 100011, China

[a]20028923@chnenergy.com.cn
[b]20037286@chnenergy.com.cn
[c*]11688090@chnenergy.com.cn

Abstract. This content discusses the importance of software interface design and the significance of visual accessibility theory and visual perception in creating effective interfaces. The paper aims to invest how these principles can be applied in software development to improve user experience and engagement. In addition, this article proposes a software interface click prediction model based on GRU with attention to improve the quality of the software interface. The experimental results on the collected data validate the effectiveness of the proposed model in this paper.

Keywords: Visual Accessibility Theory; Visual Perception; Software Development; Visual Appeal; Interface; User Experience; User Participation

1 Introduction

Effective software interface design is crucial for optimizing user experience and usability of software applications. A well-crafted interface can streamline complex tasks, minimize user errors, and boost productivity. Additionally, it can help establish brand identity and differentiate software products from competitors. As such, investing in software interface design is vital for creating successful software products that meet user expectations and needs[1].

Visual accessibility theory and visual perception are crucial factors in software interface design. A visually accessible interface is vital for users with visual impairments to effectively navigate and use the software. Meanwhile, visual perception is an essential aspect that helps designers create intuitive and user-friendly interfaces for all users. By integrating these principles into software design, developers can create products that cater to a broader audience, enhancing the overall user experience and satisfaction[2]. Therefore, designers must comprehend the significance of visual accessibility theory and visual perception in software interface design[3].

A. Rauf et al. (eds.), *Proceedings of the 3rd International Conference on Management Science and Software Engineering (ICMSSE 2023)*, Atlantis Highlights in Engineering 20,
https://doi.org/10.2991/978-94-6463-262-0_27

This paper aims to investigate how Visual Availability Theory and Visual Perception can be applied in software development. The Visual Availability Theory suggests that visual attention is limited and selective, and that visual stimuli compete for attention based on their salience and relevance. It is essential to comprehend the principles of visual perception when designing software interfaces that are user-friendly and visually attractive[4]. By utilizing these theories, software developers can improve user experience and boost user engagement. This paper will present an overview of Visual Availability Theory and Visual Perception, and explore their practical applications in software development.

2 VISUAL ACCESSIBILITY THEORY

2.1 Definition and explanation of visual accessibility theory

Visual accessibility theory refers to the concept of designing visual content in a manner that makes it easily perceivable and understandable by individuals with visual impairments. This theory involves the use of various techniques and tools, such as high-contrast color schemes, alternative text descriptions, and screen-reader compatibility, to ensure that users with visual disabilities can access and navigate through digital content without any barriers. By implementing visual accessibility principles, organizations can ensure that their products and services are inclusive and accessible to all users, regardless of their visual abilities[5].

2.2 Importance of visual accessibility theory in software interface design

Visual accessibility theory is a vital aspect of software interface design, as it guarantees that the interface can be used by individuals with visual impairments or color blindness. By integrating visual accessibility principles, designers can create interfaces that are simple to navigate and comprehend, regardless of the user's visual capabilities. This not only improves the user experience but also promotes inclusivity in software. As a result, it is crucial for designers to take visual accessibility theory into account when designing software interfaces to ensure that they are accessible to all users.

2.3 Application of visual accessibility theory in software interface design

Incorporating visual accessibility theory into software interface design is essential in developing inclusive and user-friendly products. Designers can ensure that their interfaces are accessible to users with visual impairments by applying principles such as color contrast, font size, and layout. This not only enhances the user experience but also assists companies in complying with accessibility regulations. In summary, integrating visual accessibility theory into software interface design is a crucial measure towards creating products that cater to the needs of all users.

1) Color contrast: Visual Availability Theory and Perception are crucial in creating user-friendly software interfaces. By understanding how users perceive color and

contrast, developers can design easily navigable and visually appealing interfaces. Emphasizing important information through color and contrast is essential. Understanding color perception can help avoid using difficult colors. Incorporating these principles can greatly improve the user experience.

- Customizing color settings in software interfaces for different carriers reinforces brand identity and improves user experience. Tailored color settings make navigation easier and contribute to a more user-friendly experience across all devices.
- Highlighting important information through color and contrast is crucial in software design. Visual cues improve usability, reduce errors, enhance aesthetics, and increase user engagement and retention.
- Designing user-friendly software interfaces requires understanding color and contrast perception. Opting for eye-friendly colors, ensuring sufficient contrast, and employing a consistent color scheme improves navigation and accessibility for all users.

2) Font size and style: In software development, font size and style are critical for creating user-friendly interfaces. A font that is too small or decorative can be frustrating and distracting, while a legible and aesthetically pleasing font can enhance the user's experience. Sans-serif fonts like Arial or Helvetica are commonly used, while serif fonts like Times New Roman or Georgia can be used for a more traditional feel. A font size of 10-12 points for body text is recommended, with larger sizes for headings and subheadings. Choosing the right font can lead to a positive user experience.

3) Layout and spacing: Consider visual availability theory and visual perception when designing software layouts and spacing. Visual availability theory prioritizes important information in easily accessible areas. Visual perception factors in color contrast, font size, and spacing for easy interpretation. Striking a balance between the two creates user-friendly applications. Applying these concepts is crucial for effective software design.

4) Iconography: Iconography is essential in software design as it reduces cognitive load and improves user experience by providing easily recognizable and understandable visual cues. Visual availability theory suggests that users notice and pay attention to prominent and distinctive visual elements, which can be achieved through well-designed icons. Additionally, visual perception is important in software design, and icons with a clear and consistent visual language can help users understand software functionality and navigate efficiently. Therefore, software designers should prioritize the design of consistent, visually distinct, and easily identifiable icons to enhance usability and user experience.

5) Navigation：Software development is essential in today's digital age, and developers must consider visual availability theory and visual perception. Navigation plays a key role in applying these theories, enabling users to interact with software effectively. Visual availability theory emphasizes making essential features more visible and accessible, while visual perception requires consistent design elements. Effective navigation provides clear and intuitive access to features, enhancing the user experience and creating software that is both visually appealing and user-friendly.

3 VISUAL PERCEPTION

3.1 Definition and explanation of visual perception

Visual perception is the brain's interpretation of visual information received through the eyes. It involves the detection of light and the use of cues such as color, brightness, contrast, and texture to identify objects. It is an active process influenced by past experiences, expectations, and attention. It is critical for navigating, recognizing objects and faces, and interpreting visual information, and is linked to memory, attention, and decision-making.

3.2 Importance of visual perception in software interface design

Visual perception is essential in software interface design as it helps users navigate, understand content, and perform tasks efficiently. Designers can use colors, shapes, and visual hierarchy to identify important information and create an aesthetically pleasing interface. Consistent icons and symbols can reduce the learning curve and make the software more accessible. Overall, designers must consider visual perception to create a user-friendly and engaging interface.

3.3 Application of visual perception in software interface design

1) Gestalt principles：Gestalt principles describe how humans perceive and organize visual information. They were developed by German psychologists in the early 20th century and are used in design, advertising, and psychology. There are five principles: proximity, similarity, closure, continuity, and figure-ground. These principles are used to create visual groupings, patterns, illusions, flow, and contrast. Understanding Gestalt principles is important for creating effective and visually appealing communication.

2) Fitts' Law：Fitts' Law is a principle in human-computer interaction that explains how the size and distance of targets affect the time it takes to move a cursor to them. The law states that the time required to move a cursor to a target is proportional to the distance to the target and inversely proportional to the size of the target. This means that targets should be made as large as possible and placed as close to the cursor as possible to reduce the time it takes to reach them. Targets that are frequently used should be placed in easy-to-reach areas, such as corners or edges. Fitts' Law is essential for creating user interfaces that are easy to use and efficient.

3) Hick's Law：Hick's Law states that the time it takes to make a decision increases logarithmically with the number of choices available. It was proposed by William Edmund Hick and Ray Hyman in the 1950s and has been applied in user interface design, marketing, and education. By reducing the number of choices presented to users, marketers, educators, and designers can help people make decisions more efficiently. However, it is important to remember that Hick's Law is not a universal solution and that different individuals and contexts require different approaches.

4) Miller's Law： Visual availability theory and visual perception are crucial in software design to ensure efficient user interaction. Miller's Law states that the brain can only process limited information, so interfaces should be visually appealing, easy to navigate, and present information in a clear and concise way. Visual availability theory ensures important information is easily accessible and visible, while visual perception principles like Gestalt can help users quickly identify related items. Miller's Law can also be applied by breaking complex tasks into manageable steps. These concepts create visually appealing, user-friendly interfaces that are highly functional.

5) Parkinson's Law：Parkinson's Law states that work expands to fill the time available for its completion. This highlights the importance of setting clear deadlines and managing time effectively. When given an open-ended deadline, individuals may procrastinate or take longer than necessary to complete a task. To combat this, managers should set clear expectations, monitor workloads, and encourage employees to work at a faster pace. By doing so, they can create a more productive and efficient workplace, where tasks are completed in a timely and efficient manner.

4 CASE STUDIES

4.1 Examples of software interface designs that apply visual accessibility theory and visual perception

1) High-Contrast Interfaces: These interfaces use bold, contrasting colors to make it easier for users with visual impairments to distinguish between different elements on the screen.

The interface's main color scheme is inspired by the night sky and the glow of lights, consisting of black and gold. This creates a mysterious and noble aesthetic. The dark background, combined with warm-colored data, creates a distinct contrast that makes the data stand out prominently. The design utilizes gradient colors and incorporates different tones to maintain sufficient hue and brightness differences. The dark shades enhance the vividness of the colors, ensuring that core data elements are highlighted and visually striking. This approach results in natural and seamless visualization charts and graphics.

2) Large Text and Icons: Interface designs that incorporate larger text and icons can be beneficial for users with low vision or visual impairments.

Using larger text and icons in interface designs benefits users with low vision or visual impairments, as well as older users with age-related visual impairments. This improves their experience and makes digital interfaces more accessible and user-friendly.

3) Consistent Layout and Navigation: Consistency in layout and navigation can help users with cognitive impairments or visual processing disorders to better understand and interact with the interface.

Consistent layout and navigation are crucial for a user-friendly interface. They help users with cognitive or visual impairments to easily find and access features, and navigate through the interface without confusion. Consistency is essential for creating an accessible and user-friendly interface for all users.

By applying visual accessibility theory and visual perception in software interface design, designers can create more inclusive and user-friendly interfaces that cater to a wider range of users.

4.2 Analysis of how these designs improve user experience and usability

App designs are important for improving user experience and usability. Here are some ways they can do this:

1) User-centered design:When designing an app, it's important to keep the user's needs and preferences in mind. This will ensure that the app is intuitive and easy to use, resulting in a seamless user experience. User-centered design is crucial for app development, as it prioritizes the user's needs and preferences. By doing this, developers can create functional and enjoyable apps that meet the user's expectations. Ultimately, user-centered design is about putting the user first and designing apps that cater to their needs.

2) Consistent design:Consistency in design is crucial for improving usability. Design elements like colors, fonts, and icons should be consistent throughout the app to avoid confusion and make it easier for users to navigate. When the design elements are consistent, it creates a sense of familiarity and establishes a visual hierarchy that guides users to important information and actions within the app. Inconsistencies in design can lead to confusion and frustration for users, making it harder for them to understand the content. By maintaining consistency in design, app designers can create a more intuitive and user-friendly experience for their users, leading to increased engagement, higher user satisfaction, and ultimately, more successful apps.

3) Simple and clear interface:To improve user experience and usability, app interfaces should be simple and clear, with minimal clutter. App designers can achieve this by focusing on important information and actions, using clear labels, grouping related information, and considering the context of use. By creating a user-friendly interface, app designers can increase user engagement, satisfaction, and success of the app.

4) Easy navigation:Navigation is crucial for an app's usability. Designers should ensure that users can easily navigate the app with clear labels and buttons. To achieve this, designers should create a navigation system that is easy to understand and consistent throughout the app. They should also consider the context of the app's use and optimize the navigation system accordingly. Usability testing can be used to make adjustments based on user feedback. Overall, effective navigation requires careful planning and attention to detail to create a user-friendly experience.

5) Visual feedback:Visual feedback is important to improve user experience and usability. When a user clicks a button, the button should change color or provide other visual feedback to confirm the action has been completed. Designers should choose recognizable and intuitive visual cues that are consistent throughout the app. Usability testing can evaluate the effectiveness of visual feedback.

6) Accessibility:App designs should be accessible to all users, including those with disabilities. This can be achieved by using larger fonts, providing alternative text for images, and using contrasting colors to make text easier to read. Designers should also

ensure compatibility with assistive technologies and meet accessibility standards. Usability testing with users with disabilities can help identify areas for improvement.

4.3 Comparison of current designs with those that do not apply visual accessibility theory and visual perception

Feature System Interface Design with Visual Accessibility Current Popular System Interface Design

Color Contrast Use high-contrast colors to help users distinguish between different interface elements and content Use low-contrast colors to make it easier for users to read and browse content

Font Size Use larger font sizes to make it easier for users to read content Use smaller font sizes to display more content on the screen

Image Alt Text Provide alternative text for all images so that users can understand the content of the images Do not provide alternative text, making it impossible for users to understand the content of the images

Keyboard Navigation Provide keyboard navigation so that users can navigate the application using the keyboard Do not provide keyboard navigation, forcing users to use a mouse or touch screen for navigation

Visual Feedback Provide visual feedback, such as highlighting or animation, to let users know whether their actions were successful Do not provide sufficient visual feedback, making it impossible for users to know whether their actions were successful[6].

5 Software interface click prediction

We propose an attention based GRU software interface click prediction model. Firstly, the hidden state of click preferences is calculated as follows:

$$\mathbf{R}_t = \sigma(\mathbf{X}_t\mathbf{W}_{xr} + \mathbf{H}_{t-1}\mathbf{W}_{hr} + \mathbf{b}_r),$$

$$\mathbf{Z}_t = \sigma(\mathbf{X}_t\mathbf{W}_{xz} + \mathbf{H}_{t-1}\mathbf{W}_{hz} + \mathbf{b}_z),$$

$$\tilde{\mathbf{H}}_t = \tanh(\mathbf{X}_t\mathbf{W}_{xh} + (\mathbf{R}_t \odot \mathbf{H}_{t-1})\mathbf{W}_{hh} + \mathbf{b}_h),$$

$$\mathbf{H}_t = \mathbf{Z}_t \odot \mathbf{H}_{t-1} + (1 - \mathbf{Z}_t)\tilde{\mathbf{H}}_t,$$

where $\mathbf{X}_t$ is input, $\mathbf{R}_t$, $\mathbf{Z}_t$, and $\tilde{\mathbf{H}}_t$ are reset gates, update gates and candidate hidden states respectively. $\mathbf{W}_*$ are weight parameter and $\mathbf{b}_*$ are biases. σ and tanh is activation function. We may determine the network's concealed state at time t using the formula above. The status of the previous time steps can also have an impact on the prediction outcomes, although click choices remain private. So, we present a calculational attention mechanism:

$$\beta(\boldsymbol{q}_t, \boldsymbol{k}_\tau) = \frac{\boldsymbol{q}_t^T \boldsymbol{k}_\tau}{\sqrt{d_k}},$$

$$\alpha(\boldsymbol{q}_t, \boldsymbol{k}_\tau) = \text{softmax}\,(\beta(\boldsymbol{q}_t, \boldsymbol{k}_\tau)) = \frac{\exp(\beta(\boldsymbol{q}_t, \boldsymbol{k}_\tau))}{\sum_{\tau'} \exp\,(\beta(\boldsymbol{q}_t, \boldsymbol{k}_{\tau'}))},$$

$$\boldsymbol{H}_{att} = \sum_\tau \alpha(\boldsymbol{q}_t, \boldsymbol{k}_\tau)\boldsymbol{h}_\tau,$$

A full connection layer can be used to determine the likelihood of the click prediction $\tilde{\mathbf{y}}_t$ at t once the network hidden state has been combined with the attention mechanism.

Finally, we optimize the model by minimizing the cross entropy loss:

$$\mathcal{L} = -\sum_{t=1}^{T} (\boldsymbol{y}_t \log \tilde{\mathbf{y}}_t + (1 - \boldsymbol{y}_t)\log\,(1 - \tilde{\mathbf{y}}_t))$$

where $\boldsymbol{y}_t$ is the reality of whether the software interface clicks.

6 experiment

6.1 dataset

Interface element feature extraction: Collect and extract various element features in the software interface, such as position, size, color, shape, text content, etc. These features can be extracted through interface analysis techniques or image processing algorithms. At the same time, relative relationships between elements can also be considered, such as the layout and nesting relationships of elements.

Interface element historical click data: Collect and record the historical click data of software interface elements, including information such as the number of clicks, click location, click time, etc. These data can be used to analyze and predict user click behavior on different interface elements.

Feature engineering: based on the collected interface element features and historical click data, conduct Feature engineering processing, such as feature coding, Feature selection, etc.

As shown in Figure 1, we collected the click count of 1000 software interfaces and provided the distribution of click counts.

Table 1. Performance results

Hidden Size	AUC	MAE	RMSE
16	0.7285	0.3922	0.3141
32	0.7611	0.3741	0.2953
64	0.7895	0.3547	0.2774
128	0.7933	0.3401	0.2612
256	0.8101	0.3376	0.2573
512	0.8122	0.3389	0.2551

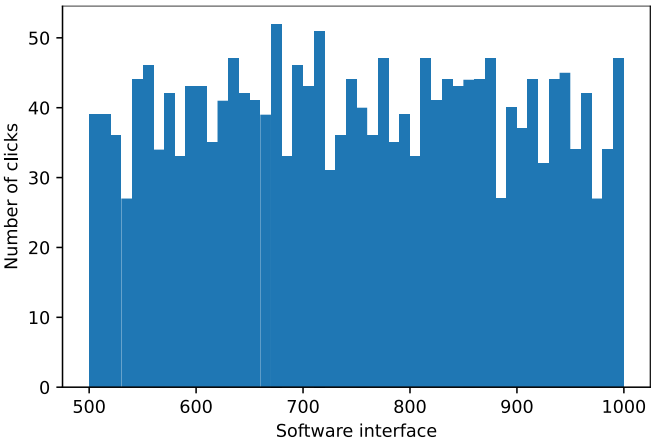

Fig. 1. Software interface click count distribution

6.2 experimental analysis

Table 1 clearly illustrates the click prediction outcomes of the software interface based on the GRU, demonstrating the value of our model for click-through rate modeling and achieving accurate prediction outcomes. Additionally, a significant hyperparameter that influences the model's prediction power and training effectiveness is the hidden layer dimension. The expressive power of the model typically rises with the size of the hidden layer, but this also lengthens training time and uses more computational resources. The size of hidden layers must therefore be assessed and optimized. We think that setting the hidden layer dimension to 256 is reasonable to guarantee the precision and effectiveness of the prediction, as shown in Table 1.

In order to conduct enough experiments, we used 30% of the data set as the test set and 70% of the data set as the training set. Figure 2 displays the training process's loss change curve. The loss of both the training set and the test set displays a declining trend as the number of iterations rises, and the training loss gradually converges.

Table 2. Performance Comparison

Model	AUC	MAE	RMSE
RNN	0.6912	0.4011	0.3233
CNN	0.6845	0.3932	0.3178
LSTM	0.7320	0.3815	0.3007
GRU	0.7480	0.3823	0.2912
Attention	0.7691	0.3698	0.2826
Att-GRU	**0.8101**	**0.3376**	**0.2573**

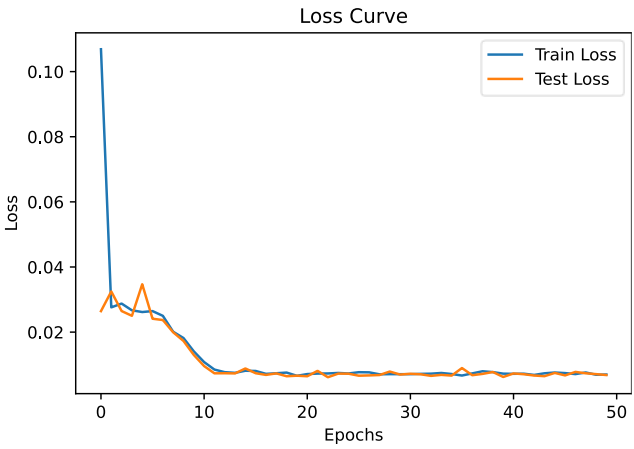

Fig. 2. The convergence curves of loss

We compared the performance of multiple models with our proposed model, as shown in Table 2, and our model showed the best performance. In addition, we can also conclude that adding attention mechanisms to GRU is beneficial for click prediction in software interfaces. In the software interface click prediction task, the performance of Long short-term memory network is also better than that of Convolutional neural network.

7 CONCLUSION

Overall, the theories of visual accessibility and visual perception play a crucial role in software interface design. Designers should understand these theories and apply them to their designs to create interfaces that are easy to access and use, thereby improving the user experience and satisfaction. In addition, we propose a software interface click prediction model based on GRU with attention, which has superior prediction performance and provides guidance for improving the quality of software interfaces.

References

1. Zhang J , Ma C , Zhong C ,et al.Multi-scale and multi-channel neural network for click-through rate prediction[J].Neurocomputing, 2022, 480:157-168.
2. Xu E , Yu Z , Guo B ,et al.Core Interest Network for Click-Through Rate Prediction[J].ACM Transactions on Knowledge Discovery from Data, 2021, 15(2):1-16.DOI:10.1145/3428079.
3. A D Z , B Z W , C L Z ,et al.Deep Field Relation Neural Network for Click-Through Rate Prediction[J].Information Sciences, 2021.
4. Yang K , Wang S , Han T ,et al.Interface optimization of La-based gate dielectric for molybdenum disulfide field-effect transistors[J].Applied Surface Science: A Journal Devoted to the Properties of Interfaces in Relation to the Synthesis and Behaviour of Materials, 2022(Apr.15):581.DOI:10.1016/j.apsusc.2021.152248.

5. Khor E T .A data mining approach using machine learning algorithms for early detection of low-performing students[J].The international journal of information and learning technology, 2022(2):39.DOI:10.1108/IJILT-09-2021-0144.
6. Hu K , Qi Y , Huang J ,et al.Continual Learning for CTR Prediction: A Hybrid Approach[J].arXiv e-prints, 2022.DOI:10.48550/arXiv.2201.06886.