CAPTCHA Unmasked: The Math That Outsmarts Bots

By

Parsa Besharat

A thesis submitted as part of the requirements for the lecture, Mathematises Seminar, of MSc Mathematics of Data and Resources Sciences at the Technische Universität Bergakademie Freiberg

Matriculation Number: 69365

February, 2024

Supervisor: Dr. Uwe Prüfert

Abstract

CAPTCHA (Completely Automated Public Turing Test to Tell Computers and Humans Apart) serves as a vital tool in securing online platforms by distinguishing human users from automated bots. This project delves into the multifaceted mechanisms of CAPTCHA, exploring its historical evolution, mathematical underpinnings, and advanced image processing techniques that bolster its effectiveness. Core topics include image distortion through affine transformations and nonlinear warping to obscure patterns, as well as color management using segmentation and noise addition to enhance complexity. The study also highlights the integration of machine learning, focusing on neural networks and convolutional operations, which extract hierarchical features from CAPTCHA images to classify inputs. Optimization strategies such as gradient descent are examined to fine-tune CAPTCHA challenges, balancing human usability with bot resistance. Despite its robust design, CAPTCHA faces challenges from evolving AI models capable of bypassing its defenses. The findings emphasize the need for adaptive and innovative approaches, such as biometric CAPTCHAs and dynamic challenges, to ensure continued effectiveness in human-bot verification systems, while addressing usability and accessibility concerns. This project consolidates mathematical rigor, image processing insights, and machine learning advancements to outline the future trajectory of CAPTCHA technology.

Contents

Introduction of CAPTCHA

1.1 Overview of CAPTCHA

CAPTCHA (Completely Automated Public Turing Test to Tell Computers and Humans Apart) is a vital security mechanism that protects online platforms by distinguishing human users from automated bots. First introduced in the early 2000s by a team at Carnegie Mellon University, CAPTCHA was designed to combat increasing misuse of digital systems, such as spamming, automated account creation, and data scraping. The basic principle behind CAPTCHA is to create challenges that are intuitive and easy for humans to solve but computationally difficult for machines. [2] Early implementations relied on distorted text and numbers, where simple visual tasks leveraged the human brain's natural ability to recognize patterns in noisy or warped environments. Over time, CAPTCHA has evolved significantly[4], incorporating more sophisticated mechanisms such as image-based challenges, audio-based tests, and behavioral analysis. These modern adaptations are powered by advancements in mathematics, image processing, and artificial intelligence, allowing CAPTCHA to remain a crucial tool in cybersecurity. [3]

However, as bots become increasingly advanced with AI capabilities, CAPTCHA systems must continually innovate to maintain their effectiveness. Despite challenges related to accessibility and usability, CAPTCHA remains a cornerstone of digital security, ensuring fair access, data integrity, and system protection across the internet.

1.2 CAPTCHA’s role in Differentiating Humans and Bots

CAPTCHA differentiates humans from bots by presenting challenges that humans can solve intuitively but are difficult for bots to process. Tasks like recognizing distorted text, selecting specific objects in images, or solving puzzles exploit human cognitive abilities while disrupting bots' reliance on structured algorithms and pattern recognition. [4] For instance, distorted text CAPTCHAs add noise and distortions that bots struggle to decode, while humans can easily recognize patterns.[1][4] Image-based CAPTCHAs, like identifying objects, leverage human visual skills to detect shapes and details under varying conditions, which bots find challenging without extensive AI training. CAPTCHA protects online platforms by preventing abuse such as spamming, data scraping, and brute-force attacks, ensuring resource fairness and system integrity. [7]

By dynamically generating challenges and validating responses, CAPTCHA remains a critical tool for maintaining secure human-bot differentiation in online systems.

1.3 How CAPTCHA works

CAPTCHA operates by presenting users with tasks that exploit the limitations of automated systems. For example, in text-based CAPTCHAs, distorted letters or numbers are generated using techniques like affine transformations and noise addition, which bots struggle to interpret. Image-based CAPTCHAs require users to identify specific objects leveraging the human ability to process and recognize visual patterns. These challenges are dynamically generated to ensure randomness and prevent bots from using pre-trained models to solve them. [1][7] CAPTCHA systems further rely on probabilistic algorithms, cryptographic randomness, and adaptive difficulty settings to balance human usability with bot resistance. [3]

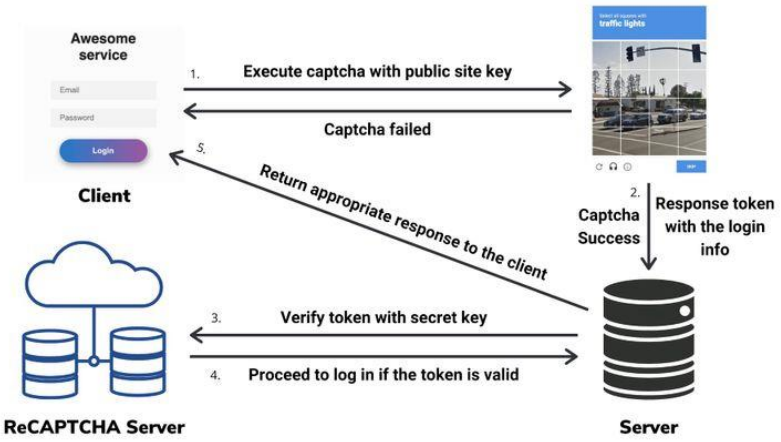

Figure 1.2.1, Illustration of How CAPTCHA Works, this diagram demonstrates the interaction between a client, server, and reCAPTCHA system during a login process. When the client accesses the login page and submits credentials, reCAPTCHA is executed using a public site key (Step 1). The client solves a CAPTCHA challenge, such as selecting traffic lights, to confirm they are human (Step 2). A response token is then sent to the server with the login information, which the server forwards to the reCAPTCHA server for verification using a secret key (Step 3). If the token is valid, the user is allowed to log in (Step 4); otherwise, the client receives a failure response (Step 5). This process ensures secure authentication by verifying users while blocking bots

1.3 Historical Context of CAPTCHA

CAPTCHA plays a vital role in distinguishing humans from bots by presenting challenges, such as recognizing distorted text or identifying objects in images, that humans can intuitively solve but bots struggle to decode. These tasks leverage human cognitive abilities while exploiting the limitations of automated systems, which rely on pattern recognition and rule-based logic bots' ability to process data efficiently. [3][5]

2. Algorithms and Mathematical Concepts

CAPTCHA's algorithms and mathematical concepts ensure challenges are human-solvable but bot-resistant, using techniques like gradient descent, cryptographic randomness, and affine transformations. Neural networks and noise addition enhance pattern recognition and security, maintaining CAPTCHA's robustness against automated attacks. [5]

2.1 Optimization: Gradient Descent for CAPTCHA

Gradient Descent is an optimization algorithm used to minimize a function by iteratively moving toward its steepest descent, defined by the negative gradient. [4]

- **Optimize Challenge Difficulty:**

  It ensures CAPTCHA challenges are solvable by humans but difficult for bots. Balance distortion, noise, and complexity. [5]

- **Train Models to Detect Bots:**

  CAPTCHA systems use machine learning models trained via gradient descent to identify patterns that differentiate bots from humans. [3]

- **Optimize CAPTCHA Parameters:**

  Parameters like noise level, distortion intensity, and image quality are fine-tuned to maximize security while preserving usability. [4]

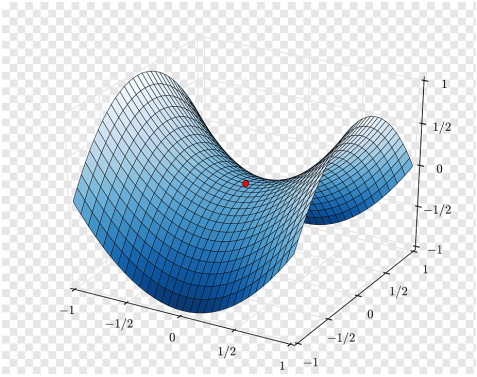

Figure 2.1.1, Illustration of Optimization: Gradient Descent for CAPTCHA, this 3D plot illustrates gradient descent for CAPTCHA, where the red dot represents the current parameter on the loss function.

2.1.1 Gradient Descent for CAPTCHA – Objective Function

Gradient descent minimizes an objective function $J(\theta)$, often the loss function, which quantifies the difference between the predicted and true values. Gradient docents allow us to calculate how each parameter contributes to the error: [4]

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} Loss\left(h_\theta\left(x^{(i)}\right), y^{(i)}\right)$$

- $h_\theta\left(x^{(i)}\right)$: *Hypothesis function (e.g., predicted CAPTCHA result).*
- $y^{(i)}$ *: True Label (Human or bot)*
- m: *Number of samples*
- *Loss: Loss function*

With respect to the parameters θ can be set as: [4]

$$\nabla_\theta J(\theta) = \frac{\partial J(\theta)}{\partial \theta}$$

This gradient points in the direction of the steepest ascent of $J(\theta)$.

2.1.2 Steps in CAPTCHA Optimization Using Gradient Descent

- **Define the Loss Function:**
  Quantify how well a CAPTCHA separates humans from bots based on prediction accuracy. [5]
- **Compute Gradients:**
  Use backpropagation to calculate gradients of the loss function with respect to CAPTCHA parameters [5]
- **Update Parameters:**
  Adjust parameters iteratively using the update rule. [5]
- **Converge to Optimal Parameters:**
  Stop when the gradient becomes small or the loss function stops decreasing significantly. [5]

2.1.3 Gradient Descent Update Rule

To minimize $J(\theta)$ the update θ iteratively by moving in the direction of the negative gradient. The Gradient Descent Update Rule is fundamental because it works to iteratively improve the parameters of a model or system by minimizing its loss function. The update rule, parameters are adjusted in the opposite direction of the gradient (steepest descent) to reduce the loss function value. [5]

$$\theta := \theta - \alpha \nabla_\theta J(\theta)$$

- α: *Learning rate, which controls the step size.*
- θ: *Parameters being optimized (e.g., distortion intensity, noise level).*

2.1.4 Gradient Descent Loss function

A loss function is essential in Optimization: Gradient Descent for CAPTCHA because it provides a measurable objective to evaluate how well the CAPTCHA system is performing. It quantifies the difference between the desired outcomes (e.g., high human success and low bot success) and the actual outcomes, guiding the optimization process. [6]

2.1.4.1 Weighted Cross-Entropy Loss

It will prioritize minimizing bot success while preserving human usability. [6]

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[w_h \,.\, y_h^{(i)} \log\left(h_\theta\left(x^{(i)}\right)\right) + w_b \,.\, y_b^{(i)} \log\left(1 - h_\theta\left(x^{(i)}\right)\right)\right]$$

- $J(\theta)$*: total loss*
- *m: Number of Captcha Instances*
- $h_\theta\left(x^{(i)}\right)$*: Model's probability output for instance i.*
- $y_h^{(i)}$ *: Label for human users (*$y_h = 1$*)*
- $y_b^{(i)}$ *: Label for human users (*$y_b = 1$*)*
- w_h *: Weight for human-related terms (emphasizes usability).*
- w_b *: Weight for human-related terms (emphasizes usability).*
- $\log(h_\theta(x))$*: Penalizes incorrect human predictions.*
- $\log(1 - h_\theta(x))$*: Penalizes incorrect bot predictions.*

Scenarios:

- ***Human Usability:***
  By assigning a higher weight $w_h > w_b$ the loss ensures that the system prioritizes making CAPTCHAs solvable for humans.

  Humans failing the CAPTCHA heavily penalizes the loss, leading to adjustments in parameters (e.g., reducing distortion or noise). [6]

- ***Minimizing Bot Success:***
  The term $w_b \, . \, y_b^{(i)} \log\left((1 - h_\theta\left(x^{(i)}\right)\right)$ penalizes the model for bot's success.

  Increasing w_b amplifies the penalty for bots solving CAPTCHAs, encouraging harder challenges. [6]

2.1.4.2 Balanced Success Rate Loss

This loss directly measures the difference between human and bot success rates, with regularization to avoid extreme difficulty. [7]

$$J(\theta) = \alpha \,.\, (1 - S_h) + \beta \,.\, S_b + \lambda \,.\, R(\theta)$$

- $J(\theta)$: total loss
- S_h *:* Human success rate (proportion of humans solving CAPTCHA).
- S_b*:* Bot success rate (proportion of bots solving CAPTCHA).
- $R(\theta)$ *:* Regularization term (penalizes excessive distortion or noise).
- α: Weight for human usability ($\alpha > \beta$)
- β*:* Weight for bot resistance.
- λ: Regularization parameter to balance usability and distortion.

Why It's Effective

1. **Human Usability**:
   The term $(1 - S_h)$ penalizes low human success rates.
   Ensures that overly distorted or noisy CAPTCHAs are avoided.
2. **Minimizing Bot Success**:
   The term S_b penalizes high bot success rates.
   Encourages harder challenges specifically targeted at bot weaknesses.
3. **Regularization**:
   The term $R(\theta)$prevents extreme distortions that might make CAPTCHAs unsolvable even for humans.

2.1.4.3 Comparison of the Two Loss Functions

Acpect	**Weighted Cross-Entropy Loss**	**Balanced Success Rate Loss**
Focus	Penalizes incorrect classification of humans and bots.	Directly evaluates human vs. bot success rates.
Complexity	Relies on probability outputs from the model.	Simpler, requires only success rate measurements.
Flexibility	Allows fine-grained tuning with W_h and W_b.	Weights α, β, λ allow holistic control.
Best Use Case	Suitable when working with predictive models (e.g., AI).	Suitable for non-AI CAPTCHA designs or high-level analysis.

2.2 Neural Networks Concepts in CAPTCHA

These systems use mathematical models to extract features from CAPTCHA images and predict whether the input belongs to a human or bot. Neural networks are mathematical models composed of layers of interconnected neurons, designed to learn patterns from data. In CAPTCHA-solving. [7]

2.2.1 Input Layer

The **input layer** in a neural network is the first layer that receives raw data and passes it to the subsequent hidden layers for processing. Each neuron in the input layer corresponds to a feature of the data, and the number of neurons matches the dimensionality of the input. For example, in a numerical dataset, each input neuron represents a specific variable, while in text data, each neuron may represent a token or word embedding. [7]

- $X = \{x_{ij} \mid i = 1, \ \ldots, m; j = 1, \ \ldots, n\}$ *, grayscale image with dimensions*
- $X \in \mathbb{R}^{m \times n \times 3}$,
  For RGB images, it becomes a 3D tensor

2.2.2 Hidden Layer

Hidden layers are the intermediate layers in a neural network located between the input and output layers. They process input data (e.g., a CAPTCHA image) and extract patterns or features that aid in solving the CAPTCHA. Each layer applies a series of transformations using weights, biases, and activation functions. Each neuron applies a weighted sum of inputs followed by an activation function to capture non-linear patterns. [7]

$$Z_i^{(l)} = \sum_j w_{ij}^{(l)} + a_j^{(l-1)} + b_i^{(l)}$$

- $z_i^{(l)}$: *The weighted input for neuron i in layer l.*
- $w_{ij}^{(l)}$ *: Weight connecting neuron j in the previous layer (l-1) to neuron i in the current layer (l).*
- $a_j^{(l-1)}$ *: The activation output of neuron j in the previous layer l-1*
- $b_i^{(l)}$ *Bias term for neuron i in the current layer l*

What Hidden Layers Do?

- The first hidden layer detects edges, curves, and corners in the image.
- Deeper layers extract more abstract features, such as letter shapes or numeric patterns.
- Outcome: Hidden layers progressively break down the image into a hierarchy of features, allowing the network to focus on meaningful elements.

2.2.3 Activation Function [7]

An activation function is a mathematical function applied to the output of a neuron in a neural network. It introduces non-linearity into the network, allowing it to learn and model complex patterns in data. Introduces non-linearity, enabling the network to solve complex problems like distorted CAPTCHA images. [6][7]

2.2.3.1 Comparison of the Two Loss Functions

Feature	ReLU	Balanced Success Rate Loss
Formula	$a = \max(0, z)$	$a = \frac{1}{1 + e^{-z}}$
Efficency	Fast computation (simple comparison)	Slower (exponential computation)
Range	$[0,\infty)$	$(0,1)$
Gradient Behavior	Constant for positive inputs 1	Diminishing gradients for large
Non-Linearity	Linear for positive, zero for negative inputs	Smooth non-linearity

2.3 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are essential in CAPTCHA-solving due to their ability to extract features from visual data, such as distorted text or image-based challenges. By using convolution operations to detect edges, curves, and patterns, and pooling layers to reduce noise and focus on critical features, CNNs effectively process and classify CAPTCHA inputs. Their hierarchical learning approach makes them robust against distortions and noise, enabling accurate recognition of complex CAPTCHA patterns. [5][6]

2.2.1 Convolution Operation [5][6]

It captures local features (edges, corners, distortions) by sliding filters (kernels) over the image.

$$(I \times K)(x, y) = \sum_i \sum_j I(x + i, y + j).\ K(i, j)$$

for an Image *I* and Filter *K*.

Where:

- $I(x + i, y + j)$*: Pixel value of the input at position of (x+i,y+j)*
- $K(i, j)$*: Value of the kernel at position (i,j)*
- K*: Size of the kernel*
- *(x,y): Position of the filter on the input matrix.*

Why It's Crucial for CAPTCHA[5][6]

In CAPTCHA-solving, the convolution operation detects essential patterns like edges, curves, or object boundaries, even in distorted or noisy images. This hierarchical feature extraction enables the CNN to understand complex inputs and classify CAPTCHA challenges accurately.

2.2.2 Pooling [1][2][5][6]

Reduces the spatial dimensions of the feature maps, preserving key features while discarding irrelevant data (like noise):

$$P(x, y) = \max\left(I_{x:x+k,y:y+k}\right),$$

Max-Pooling: Takes the maximum value in a sliding window

$$P(x, y) = \frac{1}{k^2}\sum_{i=1}^{k}\sum_{j=1}^{k} I(x + i, y + j),$$

Average Pooling: Takes the average of values in a sliding window

2.2.3 Feature Maps [1][2][5][6]

feature map is the output generated after applying convolution operations to an input image. It highlights essential patterns like edges, curves, textures, or regions of interest, making it easier for the neural network to process distorted or noisy CAPTCHA data. Each feature map corresponds to a specific filter (kernel) applied during convolution, extracting distinct features such as horizontal lines, vertical lines, or specific shapes. For example, in a distorted text CAPTCHA, feature maps may focus on the edges of letters, while in image-based CAPTCHAs, they may identify objects like traffic lights or stop signs.

3. CAPTCHA's Image Processing Concepts [1][2][5][6]

Image processing in CAPTCHA leverages various techniques to obscure patterns, enhance security, and make challenges difficult for bots while remaining solvable for humans. Core concepts include image distortion, where transformations like affine warping and nonlinear twisting alter the geometry of text or objects to confuse automated recognition. Noise addition, such as random pixel patterns or grid lines, further complicates the task for bots by masking critical features. Color management involves manipulating color spaces and adding randomness to make pattern detection harder. Techniques like thresholding simplify image complexity, while morphological operations (e.g., dilation, erosion) refine shapes by merging or breaking apart regions. Together, these methods ensure CAPTCHAs are resistant to automated attacks by exploiting the visual adaptability of humans and challenging the rigid algorithms bots rely on. [1][2][5][6]

3.1 Image Distortion

Image distortion is a crucial technique in CAPTCHA design to make text or images harder for bots to interpret while remaining solvable for humans. Distortion transforms the geometric or spatial properties of the image. [3][4]

3.1.1 Affine Transformation [3][4]

Affine transformations apply linear mapping to preserve points, straight lines, and planes while changing the scale, position, or orientation.

$$T(x,y) = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix}$$

Where:

- *(x,y): Original pixel coordinates.*
- *T(x,y): Transformed pixel coordinates.*
- *(a,d): Scaling factors.*
- *(b,c): Shear factors.*
- *(e,f): Translation offsets.*

Scenario:

- Distorted Text CAPTCHA: Stretching or compressing the text horizontally and vertically by adjusting *a* and *d*.
- Tilting the text with shear parameters *b* and *ccc*.
- Translating the text by modifying *e* and *f*.

3.1.2 Nonlinear Warping [3][4]

- Nonlinear warping applies a non-linear function to pixel positions to create irregular distortions. Sine and cosine distortions make letters wavy, preventing bots from accurately identifying text shapes.

$$T(x,y) = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x + \alpha \sin(\beta y) \\ y + \gamma \cos(\delta x) \end{bmatrix}$$

Where:

- *x′,y′: Transformed coordinates*
- *α,β,γ,δ: Warping parameters controlling intensity and frequency of the distortion. Trigonometric functions create wave-like distortions.*

3.1.3 Inverse Mapping

It reverses the transformation to restore the original image. Also, it is used to verify CAPTCHA decoding by bots.

$$T^{-1}(x',y') = \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} \begin{bmatrix} x' \\ y' \end{bmatrix} - \begin{bmatrix} e \\ f \end{bmatrix}$$

3.2 Color Management

Color management in CAPTCHA leverages color manipulation to obscure patterns, introduce noise, and create distractions that challenge automated bots. A key technique is color space conversion, which transforms images between spaces like RGB, HSV, or grayscale to modify color properties and increase complexity.

3.2.1 Converting RGB to grayscale

RGB to grayscale conversion simplifies CAPTCHA images by reducing the three-color channels (Red, Green, and Blue) into a single intensity channel while preserving the perceived brightness of the image. This is achieved using the formula:

$$I_{grey} = 0.2989R + 0.5870G + 0.1140B$$

where:
R, G, and B represent the intensities of the red, green, and blue channels, respectively.

The weights reflect the human eye's sensitivity to each color, with green contributing the most to brightness perception. Grayscale conversion removes color complexity, allowing CAPTCHA systems to focus on patterns and structures that are easily interpreted by humans but challenging for bots.

3.2.2 RGB to HSV Conversion

RGB to HSV conversion transforms an image from the Red-Green-Blue (RGB) color space into the Hue-Saturation-Value (HSV) color space, separating color information from brightness. In this conversion, Hue (H) represents the type of color, Saturation (S) measures the intensity or vividness of the color, and Value (V) indicates brightness.

The Saturation and Value are calculated using:

$$H = cos^{-1}\left(\frac{\frac{1}{2} \cdot [(R-G)+(R-B)]}{\sqrt{(R-G)^2+(R-G)(G-B)}}\right)$$

$$S = \frac{max(R,G,B) - \min(R,G,B)}{max(R,G,B)},$$

$$V = max(R,G,B)$$

Where:

- *Max(R,G,b): Maximum Intensity*
- *Min(R,G,B): Minimum Intensity*

This conversion enables CAPTCHA designers to manipulate specific color properties, such as randomizing brightness or hue, making it more difficult for bots to recognize patterns.

Significance in CAPTCHA

The flexibility of these conversions allows CAPTCHA systems to obscure visual patterns effectively. Grayscale conversion simplifies the visual content, focusing on structural details, while HSV conversion enables dynamic manipulation of colors and brightness. For instance, randomizing *S* (saturation) or *V* (value) can make the CAPTCHA appear visually diverse, while humans can still adapt to the changes.

3.2.3 Color Segmentation [5]

Color segmentation divides an image into regions based on specific color properties, such as grouping similar colors or isolating certain objects. In CAPTCHA, this is often used for object-selection tasks like "Select all traffic lights," where the CAPTCHA relies on color to differentiate objects. By applying segmentation techniques, CAPTCHA systems can highlight specific areas of interest while randomizing or altering shades of color to confuse bots. Bots struggle to identify objects due to inconsistent color patterns introduced by segmentation, while humans can use their natural perception of shapes and color relationships to complete the task successfully. [5]

$$S(x,y) = \begin{cases} 1 & if\ C(x,y) \in Target\ Color\ Range \\ 0 & otherwise \end{cases}$$

This formula defines a segmented mask *S(x,y)* that identifies whether a pixel at position *(x,y)* belongs to a specific target color range. [5]

3.2.4 Noise Addition in Color Channels

Noise addition involves introducing random variations in the color channels of an image to disrupt bots' ability to accurately analyze or segment CAPTCHA content. By adding noise to the red, green, and blue channels, CAPTCHA creates visually chaotic backgrounds that obscure the objects or text. For example, Gaussian noise spreads random color disturbances across the image, masking patterns and making it harder for bots to detect and classify regions. Humans, however, can adapt to this visual disruption by focusing on the prominent features, ensuring the CAPTCHA remains solvable for legitimate users while posing significant challenges for automated systems. [5]

$$C'(x,y) = C(x,y) + N(0,\sigma^2)$$

This formula introduces Gaussian noise into the color channel of an image.

- $C'(x,y)$: Represents the new color value at position (x,y) after adding noise.
- $C(x,y)$: Original color value at position (x,y)
- $N(0,\sigma^2)$: Gaussian noise with a mean of 0 and variance σ^2.

Scenario**:**

The Gaussian noise adds random variations to the pixel's color value, making the image visually noisy and harder for bots to analyze.

The parameterσ^2 controls the intensity of the noise:

- A larger σ^2introduces more randomness and visual disruption.
- A smaller σ^2adds subtle noise, preserving overall image clarity.

This technique is used to create noisy backgrounds or distort objects, masking patterns from bots while remaining identifiable to humans. For example, adding noise to the red, green, and blue channels makes object detection and segmentation difficult for automated systems. [3] [5]

4. Conclusion

CAPTCHA remains a vital tool in online security, distinguishing humans from bots to protect digital platforms from abuse such as spamming, data scraping, and brute-force attacks. By leveraging mathematical concepts, image processing techniques, and machine learning, CAPTCHA has evolved to remain effective against increasingly sophisticated AI-driven bots. [7] Techniques like color manipulation, noise addition, and convolutional neural networks ensure that CAPTCHAs are robust yet solvable by humans.[1][2] Despite challenges like accessibility issues and advanced bot capabilities, CAPTCHA continues to adapt, integrating innovative solutions to balance security, usability, and inclusivity. Its ongoing evolution ensures its relevance as a key defense mechanism in the ever-changing landscape of digital threats. [3]

5. References

[1] Luis von Ahn, Manuel Blum, Nicholas J. Hopper, and John Langford "CAPTCHA: Using Hard AI Problems for Security" *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, 2003

[2] Goodfellow, Ian, et al *Deep Learning* MIT Press, 2016 (Chapter on Convolutional Neural Networks)

[3] O'Shaughnessy, David *Introduction to Image Processing* Springer, 2014 (Sections on Affine Transformations and Color Management)

[4] Gonzalez, Rafael C., and Richard E. Woods *Digital Image Processing* Pearson, 2018 (Sections on Image Transformations and Noise Addition)

[5] Szeliski, Richard *Computer Vision: Algorithms and Applications* Springer, 2020 (Chapters on Image Segmentation and Feature Detection)

[6] Bishop, Christopher M. *Pattern Recognition and Machine Learning* Springer, 2006 (Chapters on Optimization and Neural Networks)

[7] Aggarwal, Charu C. *Neural Networks and Deep Learning: A Textbook* Springer, 2018 (Focus on practical deep learning applications like CAPTCHA)