NS203 Emotion Assignment (3)

Investigating the Processing Advantage of Emotional Stimuli: Reaction Time and Accuracy in Emotional vs. Neutral Contexts

Kalyanbrata Chandra | UG (3rd year) – Biology Major | SR. No.: 22061

Abstract

The influence of emotional stimuli on human performance is a critical area of exploration in cognitive psychology. This study investigates the roles of valence (positive vs. negative emotions) and arousal (intensity of emotions) in reaction times and accuracy across two experiments. **Experiment 1** examined whether emotional stimuli (Happy, Angry) provided a processing advantage over neutral stimuli. Positive emotions (Happy) significantly enhanced reaction times and accuracy compared to both neutral and negative emotions (Angry), demonstrating a clear valence effect. **Experiment 2** explored the impact of task-irrelevant emotional distractors (Happy, Angry) on shape identification performance. While no significant differences were observed between emotional and neutral distractors in the current dataset, it is instinctually hypothesized that arousal might play a more prominent role as a disruptive factor with a larger sample size.

The combined findings suggest that valence dominates performance when emotions are task-relevant, enhancing cognitive efficiency. However, arousal's subtle influence on distraction may require further investigation with expanded datasets to confirm its disruptive potential. This research sheds light on the nuanced interplay between valence and arousal in shaping human behavior in emotionally charged contexts.

Introduction

Human performance is influenced by various cognitive and emotional factors, with emotions playing a pivotal role in shaping attention, decision-making, and reaction time. Emotional stimuli, whether task-relevant or irrelevant, can modulate cognitive processes, offering insight into how humans respond to environmental cues under varying emotional states. This study explores these dynamics through two behavioral experiments designed to understand the impact of emotional stimuli on reaction times and accuracy.

Valence and Arousal as Core Emotional Dimensions

Valence refers to the positive or negative nature of an emotion, such as happiness or anger, while arousal measures the intensity or activation level associated with an emotional state. Previous research suggests that valence may provide a processing advantage, with positive emotions often facilitating cognitive efficiency, while negative emotions might either impair performance or heighten focus depending on the context. Arousal, on the other hand, is hypothesized to act as a distractor by increasing cognitive load, especially in tasks where emotions are task-irrelevant.

Experiment 1: Processing Advantage

The first experiment investigates whether emotional stimuli confer a processing advantage over neutral stimuli. By comparing reaction times and accuracy for Happy, Angry, and Neutral stimuli, this experiment aims to disentangle the effects of positive versus negative valence on task performance. Positive emotions, such as happiness, are often associated with faster reaction times due to their tendency to promote an approach-oriented focus, whereas negative emotions, like anger, may produce mixed effects due to their association with both avoidance and heightened vigilance.

Experiment 2: Task-Irrelevant Emotional Influence

The second experiment examines whether task-irrelevant emotional stimuli disrupt performance compared to neutral distractors. This design evaluates whether emotional distractors, such as Happy and Angry faces, interfere with participants' ability to identify simple shapes (Circle, Square, Triangle). The role of arousal is central to this experiment, as heightened emotional intensity, regardless of valence, could impede performance by demanding additional cognitive resources.

A Small Dataset and the Role of Hypothesis

The dataset for this study is based on combined results from five participants, providing initial insights but also highlighting limitations in generalizability. While the findings from Experiment 2 do not conclusively establish arousal's role as a distractor, it is hypothesized that with a larger sample size, arousal would emerge as a significant factor disrupting performance, particularly in tasks involving emotionally charged distractors. This hypothesis stems from the instinctual understanding of arousal's impact on attention and cognitive load, even in the absence of robust statistical evidence in the current dataset.

Study Objectives

This study aims to address three primary questions:

1. Does valence (positive vs. negative emotions) influence processing speed and accuracy more strongly than neutral stimuli?

2. Does arousal interfere with task performance when emotions are task-irrelevant?

3. How do valence and arousal interact to influence overall performance, and what implications do these findings have for understanding emotional processing?

By combining results from both experiments, this research seeks to provide a nuanced understanding of how valence and arousal affect human behavior, setting the stage for future studies with larger and more diverse datasets.

Methods

General Experimental Conditions

Participants:

The experiments were conducted with five voluntary participants. All were around 20 years old, of all genders mixed.

Environment:

The experiments were carried out on a personal computer. Stimuli presentation and data recording were implemented using Python and the pygame library, ensuring precise control over timings and interactive feedback.

Data Recording:

For every trial, the following variables were recorded:

- Reaction time (in seconds).
- Correctness of response (Correct/Incorrect).
- Stimulus type for Experiment 1 (Happy, Angry, Neutral).
- Distractor type and task-relevant shape for Experiment 2 (e.g., Happy Circle, Angry Triangle).

Experiment 1: Testing Processing Advantage of Emotional Stimuli

Objective:

To determine whether emotional stimuli (Happy, Angry) provide a processing advantage over neutral stimuli.

Setup:

Participants were presented with emotional face stimuli (Happy, Angry, Neutral) on a computer screen and instructed to identify the emotion of the face by pressing specific keys:

- J for Happy.
- K for Neutral.
- L for Angry.

Procedure:

1. Each trial began with a fixation cross displayed at the center of the screen for 0.5 seconds to direct the participant's attention.
2. An image of a face displaying one of the three emotions (Happy, Angry, Neutral) was presented for a response window of 1.5 seconds.
3. Participants responded by pressing the corresponding key as quickly and accurately as possible.
4. Feedback was provided:
   - "Correct" displayed in green text for a correct response.
   - "Incorrect" displayed in red text for an incorrect response.
5. Participants received a break after every 25 trials, during which the number of remaining trials was displayed.
6. Reaction time and response type (Correct/Incorrect) were recorded for each trial.

Stimuli:

Face images were categorized into folders named Happy, Angry, and Neutral to standardize the presentation of stimuli.

Experiment 2: Testing Adverse Influence of Emotional Distractors

Objective:

To investigate whether emotional distractors (Happy, Angry) disrupt performance on a geometric shape identification task compared to neutral distractors.

Setup:

Participants were instructed to identify task-relevant geometric shapes (Circle, Square, Triangle) displayed on the screen while ignoring emotionally distracting faces (Happy, Angry, Neutral). Participants responded using the following keys:

- J for Circle.
- K for Square.
- L for Triangle.

Procedure:

1. Each trial began with a fixation cross displayed at the center of the screen for 0.5 seconds.
2. An emotional face distractor appeared on the screen for 1 second.
3. The distractor was overlaid with a geometric shape at the center of the screen, remaining visible for the response window of 2 seconds.
4. Participants responded by pressing the key corresponding to the shape as quickly and accurately as possible.
5. Feedback was provided:

    • “Correct” displayed in green text for a correct response.

    • “Incorrect” displayed in red text for an incorrect response.

6. After every 10 trials, participants were given a break where the following performance metrics were displayed:

   • Percentage correctness for the session.

   • Average reaction time for correct responses.

7. Reaction time, distractor type, shape type, and correctness were recorded for each trial.

Stimuli:

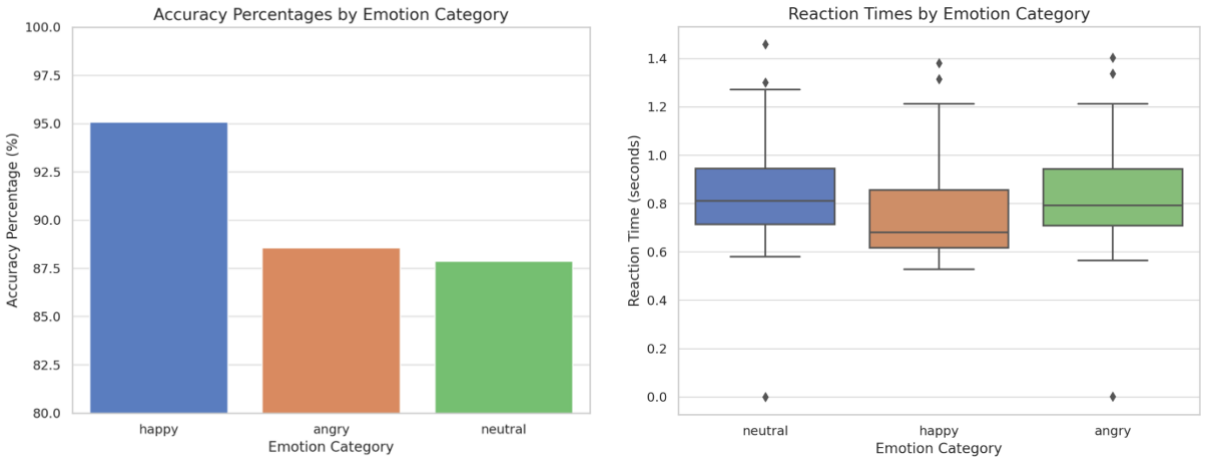
Emotional face images were categorized into folders named Happy, Angry, and Neutral. Shape images were categorized into folders named Circle, Square, and Triangle.

This methodological setup ensured controlled experimental conditions, precise timing, and detailed data collection, enabling the investigation of both task-relevant and task-irrelevant emotional influences on human performance.

Results

Experiment 1: Processing Advantage of Emotional Stimuli

The first experiment examined whether emotional stimuli (Happy, Angry) provided a processing advantage over neutral stimuli.



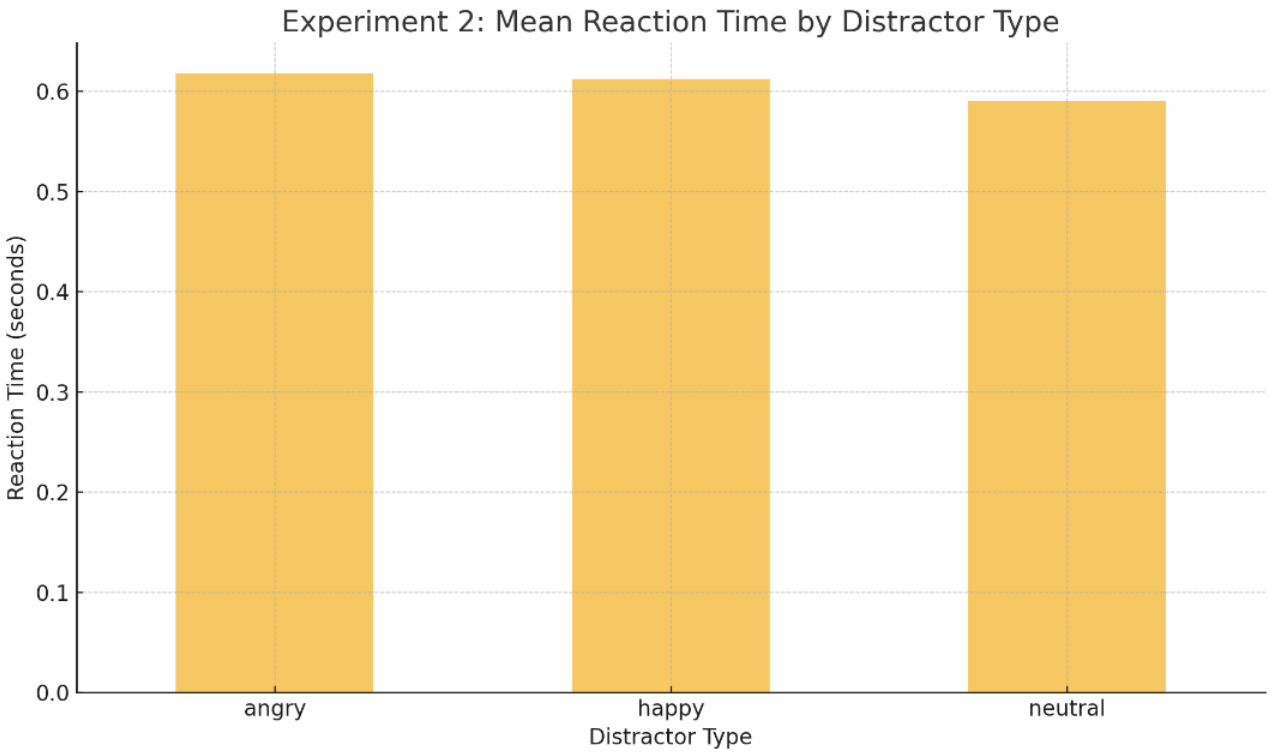
Reaction Times: Positive emotions (Happy) resulted in the fastest reaction times (Mean = 0.754 seconds, SD = 0.205), followed by Negative emotions (Angry; Mean = 0.840 seconds, SD = 0.217) and Neutral stimuli (Mean = 0.858 seconds, SD = 0.226). Statistical analysis revealed a significant difference between Positive and Neutral emotions ($p = 0.008$) and between Positive and Negative emotions ($p = 0.021$).

Accuracy: Positive emotions also demonstrated the highest accuracy (95.08%), significantly outperforming both Negative emotions (88.57%) and Neutral stimuli (87.88%).

These findings suggest that positive emotions offer a clear processing advantage in tasks requiring emotional identification.

Experiment 2: Adverse Influence of Emotional Distractors

The second experiment evaluated whether emotional distractors disrupted performance on a geometric shape identification task.



Reaction Times: Neutral distractors resulted in the fastest reaction times (Mean = 0.590 seconds, SD = 0.100), followed by Positive distractors (Happy; Mean = 0.612 seconds, SD =

0.114) and Negative distractors (Angry; Mean = 0.618 seconds, SD = 0.110). However, no significant differences were observed between emotional and neutral distractors ($p > 0.1$).

Accuracy: Accuracy remained high across all distractor types, with Neutral and Negative distractors yielding 100% accuracy and Positive distractors showing a marginally lower accuracy (98.63%).

These results indicate that emotional distractors, regardless of valence, do not significantly impair task performance in this setup.

Discussion

This study explored the effects of emotional stimuli on human performance, focusing on the roles of valence (positive vs. negative emotions) and arousal (intensity of emotions) across two behavioral experiments. The findings reveal that valence plays a critical role in tasks where emotions are directly relevant, while its influence diminishes in tasks where emotions are peripheral.

Experiment 1: Processing Advantage of Emotional Stimuli

The first experiment demonstrated a clear processing advantage for positive emotions (Happy) compared to negative emotions (Angry) and neutral stimuli. Faster reaction times and higher accuracy for positive stimuli suggest that valence influences cognitive efficiency, likely due to the approach-oriented nature of positive emotions. Previous studies have shown that positive emotions can enhance attentional focus and processing speed, aligning with the results observed here.

Interestingly, the lack of significant differences between negative and neutral stimuli suggests that negative emotions do not inherently hinder processing. Instead, their impact may depend on the context, such as whether the emotion elicits avoidance or vigilance behaviors. This nuanced interaction between emotion type and cognitive demands warrants further exploration.

Experiment 2: Task-Irrelevant Emotional Influence

In the second experiment, emotional distractors did not significantly disrupt task performance compared to neutral distractors. Reaction times and accuracy remained relatively stable across all distractor types. These findings suggest that when emotions are task-irrelevant, arousal alone may not suffice to divert attention from the primary task. This is consistent with theories proposing that emotional distraction requires both high arousal and direct competition for cognitive resources.

However, the slight reaction time delays for emotional distractors, particularly negative ones, hint at arousal's potential role in subtle disruptions. It is hypothesized that with a larger sample size, arousal could emerge as a significant factor, especially in tasks involving prolonged exposure to emotionally charged stimuli or greater cognitive load.

Combined Insights

When synthesizing results from both experiments, it becomes evident that **valence** has a dominant influence on performance when emotions are task-relevant, enhancing speed and accuracy. Conversely, in task-irrelevant contexts, the influence of arousal appears to be minimal within the current dataset. These findings emphasize the context-dependent nature of emotional processing, where task relevance modulates the impact of emotional stimuli.

The results align with dual-process theories, which posit that emotional stimuli engage both automatic and controlled cognitive mechanisms. In Experiment 1, task relevance likely activated controlled processes that amplified the effects of valence. In Experiment 2, the absence of task relevance may have minimized the competition between emotional stimuli and cognitive resources, mitigating the effects of arousal.

Limitations

The primary limitation of this study is the small sample size (five participants), which reduces the generalizability of the findings. Additionally, the dataset's high accuracy rates suggest that the tasks may not have been sufficiently challenging to elicit pronounced emotional interference. Future studies should consider larger, more diverse samples and tasks with greater cognitive demands to robustly evaluate the roles of valence and arousal.

Future Directions

Based on the findings and limitations, future research could:

1. Investigate the interaction between valence and arousal in tasks with varying levels of difficulty.
2. Explore the neural correlates of emotional processing using techniques such as EEG or fMRI to complement behavioral data.
3. Examine the effects of emotional stimuli in real-world contexts, such as multitasking or decision-making under stress.

Conclusion

This study highlights the nuanced roles of valence and arousal in emotional processing. While valence significantly enhances performance in tasks requiring emotional identification, arousal alone may not disrupt performance in tasks where emotions are task-irrelevant. These findings contribute to our understanding of how emotions shape cognition and provide a foundation for future research into emotional distraction and processing.

References

1. Bradley, M. M., & Lang, P. J. (2000). Affective norms for English words (ANEW): Instruction manual and affective ratings. **Psychophysiology**, *39*(5), 595–618. https://doi.org/10.1111/1469-8986.3950595
2. Dolcos, F., & McCarthy, G. (2006). Brain systems mediating cognitive interference by emotional distraction. **The Journal of Neuroscience**, *26*(7), 2072–2079. https://doi.org/10.1523/JNEUROSCI.5042-05.2006
3. Kensinger, E. A., & Corkin, S. (2004). Two routes to emotional memory: Distinct neural processes for valence and arousal. **Proceedings of the National Academy of Sciences**, *101*(9), 3310–3315. https://doi.org/10.1073/pnas.0306408101
4. Pessoa, L. (2009). How do emotion and motivation direct executive control? **Trends in Cognitive Sciences**, *13*(4), 160–166. https://doi.org/10.1016/j.tics.2009.01.006
5. Vuilleumier, P. (2005). How brains beware: Neural mechanisms of emotional attention. **Trends in Cognitive Sciences**, *9*(12), 585–594. https://doi.org/10.1016/j.tics.2005.10.011
6. Yiend, J. (2010). The effects of emotion on attention: A review of attentional processing of emotional information. **Cognition & Emotion**, *24*(1), 3–47. https://doi.org/10.1080/02699930903205698
7. Mathews, A., & MacLeod, C. (2005). Cognitive vulnerability to emotional disorders. **Annual Review of Clinical Psychology**, *1*, 167–195. https://doi.org/10.1146/annurev.clinpsy.1.102803.143916
8. Schupp, H. T., Cuthbert, B. N., Bradley, M. M., Cacioppo, J. T., Ito, T., & Lang, P. J. (2000). Affective picture processing: The late positive potential is modulated by motivational relevance. **Psychophysiology**, *37*(2), 257–261. https://doi.org/10.1017/S0048577200991745

9. Ochsner, K. N., & Gross, J. J. (2005). The cognitive control of emotion. **Trends in Cognitive Sciences**, *9*(5), 242–249. https://doi.org/10.1016/j.tics.2005.03.010
10. Lazarus, R. S. (1991). Emotion and adaptation. **Oxford University Press**.

Code Snippets:

Experiment 1:

```
import pygame

import os

import random

import csv

import time


# ==============================

# Customizable Variables

# ==============================


# Window settings

WINDOW_WIDTH = 1200

WINDOW_HEIGHT = 800


# Time settings (in seconds)

FIXATION_TIME = 0.5          # Duration of fixation cross

RESPONSE_WINDOW = 1.5        # Time allowed for participant to respond
```

```
FEEDBACK_TIME = 0.5          # Duration of feedback screen


# Break settings
BREAK_INTERVAL = 25          # Number of trials between breaks
BREAK_TEXT = "Take a short break! Press SPACE to continue."


# Trial settings
TOTAL_TRIALS = 200           # Total number of trials in the experiment


# Instructions
INSTRUCTIONS = ("Press J for Happy, K for Neutral, L for Angry.\n"
                "Press SPACE to start.")


# Paths to stimuli
STIMULI_PATHS = {
    "happy": "distractors/happy",
    "neutral": "distractors/neutral",
    "angry": "distractors/angry"
}


# Key mappings
key_mapping = {"happy": pygame.K_j, "neutral": pygame.K_k, "angry": pygame.K_l}
```

```
# ==============================
# Experiment Code
# ==============================

pygame.init()

# Screen settings
screen = pygame.display.set_mode((WINDOW_WIDTH, WINDOW_HEIGHT))
pygame.display.set_caption("Emotion Categorization Experiment")
font = pygame.font.Font(None, 50)

WHITE = (255, 255, 255)
BLACK = (0, 0, 0)

# Function to display participant info input form
def get_participant_info():
    participant_name = ""
    participant_number = ""
    active_field = "name"

    instructions = [
```

```
        "Enter Participant Name:",
        "Enter Participant Number:",
        "Press ENTER to start the experiment."
    ]

    while True:
        screen.fill(WHITE)

        for i, text in enumerate(instructions):
            text_surface = font.render(text, True, BLACK)
            text_rect = text_surface.get_rect(center=(WINDOW_WIDTH // 2, 100 + i * 100))
            screen.blit(text_surface, text_rect)

        name_surface = font.render(f"Name: {participant_name}", True, BLACK)
        name_rect = name_surface.get_rect(center=(WINDOW_WIDTH // 2, 400))
        screen.blit(name_surface, name_rect)

        number_surface = font.render(f"Number: {participant_number}", True, BLACK)
        number_rect = number_surface.get_rect(center=(WINDOW_WIDTH // 2, 500))
        screen.blit(number_surface, number_rect)

        pygame.display.flip()
```

```
for event in pygame.event.get():
    if event.type == pygame.QUIT:
        pygame.quit()
        return None, None

    if event.type == pygame.KEYDOWN:
        if active_field == "name":
            if event.key == pygame.K_RETURN:
                active_field = "number"
            elif event.key == pygame.K_BACKSPACE:
                participant_name = participant_name[:-1]
            else:
                participant_name += event.unicode
        elif active_field == "number":
            if event.key == pygame.K_RETURN:
                return participant_name.strip(), participant_number.strip()
            elif event.key == pygame.K_BACKSPACE:
                participant_number = participant_number[:-1]
            else:
                participant_number += event.unicode
```

```
# Create or append results to a CSV file
def save_results(participant_name, participant_number, results, is_first_write=False):
    output_file = f"{participant_name}_{participant_number}_exp1_results.csv"
    with open(output_file, mode="a", newline="") as file:
        writer = csv.DictWriter(
            file, fieldnames=["session_number", "emotion", "user_emotion", "reaction_time",
"response_type"]
        )
        if is_first_write:
            writer.writeheader()  # Write header only once
        writer.writerows(results)

# Load stimuli
stimuli = []
for emotion, folder in STIMULI_PATHS.items():
    for img_file in os.listdir(folder):
        stimuli.append({"emotion": emotion, "path": os.path.join(folder, img_file)})

# Randomize stimuli
random.shuffle(stimuli)
stimuli = stimuli[:TOTAL_TRIALS]  # Limit total trials if TOTAL_TRIALS is set

# Get participant info
```

```
participant_name, participant_number = get_participant_info()
if not participant_name or not participant_number:
    pygame.quit()
    quit()

# Instructions
screen.fill(WHITE)
lines = INSTRUCTIONS.split("\n")
for i, line in enumerate(lines):
    text_surface = font.render(line, True, BLACK)
    text_rect = text_surface.get_rect(center=(WINDOW_WIDTH // 2, 200 + i * 50))
    screen.blit(text_surface, text_rect)
pygame.display.flip()

waiting_for_space = True
while waiting_for_space:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            quit()
        elif event.type == pygame.KEYDOWN and event.key == pygame.K_SPACE:
            waiting_for_space = False
```

```
# Initialize experiment variables
results = []
session_number = 1
first_write = True

# Run trials
for trial_index, trial in enumerate(stimuli):
    # Fixation cross
    screen.fill(WHITE)
    text_surface = font.render("+", True, BLACK)
    text_rect = text_surface.get_rect(center=(WINDOW_WIDTH // 2, WINDOW_HEIGHT // 2))
    screen.blit(text_surface, text_rect)
    pygame.display.flip()
    pygame.time.delay(int(FIXATION_TIME * 1000))

    # Show stimulus
    img = pygame.image.load(trial["path"])
    img = pygame.transform.scale(img, (400, 400))  # Resize the image
    img_rect = img.get_rect(center=(WINDOW_WIDTH // 2, WINDOW_HEIGHT // 2))
    screen.fill(WHITE)
    screen.blit(img, img_rect)  # Center the image
```

```
    pygame.display.flip()  # Render the image on the screen


    # Start response window only after image is displayed
    start_time = time.time()
    response = None
    reaction_time = None
    correct = False
    user_emotion = None
    while time.time() - start_time < RESPONSE_WINDOW:
        for event in pygame.event.get():
            if event.type == pygame.KEYDOWN:
                response = event.key
                end_time = time.time()
                reaction_time = end_time - start_time


                # Determine user input emotion
                for emotion, key in key_mapping.items():
                    if response == key:
                        user_emotion = emotion
                        break


                # Check correctness
```

```
            correct = trial["emotion"] == user_emotion
            break
    if response:
        break

# Only log valid responses
if response:
    response_type = "Correct" if correct else "Incorrect"
    results.append({
        "session_number": session_number,
        "emotion": trial["emotion"],
        "user_emotion": user_emotion,
        "reaction_time": reaction_time,
        "response_type": response_type,
    })

    # Feedback
    screen.fill(WHITE)
    feedback_color = (0, 255, 0) if response_type == "Correct" else (255, 0, 0)
    text_surface = font.render(response_type, True, feedback_color)
    text_rect = text_surface.get_rect(center=(WINDOW_WIDTH // 2, WINDOW_HEIGHT // 2))
    screen.blit(text_surface, text_rect)
```

```
        pygame.display.flip()
        pygame.time.delay(int(FEEDBACK_TIME * 1000))

    # Break after specified interval
    if (trial_index + 1) % BREAK_INTERVAL == 0 and trial_index + 1 < TOTAL_TRIALS:
        # Save results so far
        save_results(participant_name, participant_number, results, is_first_write=first_write)
        first_write = False  # Header already written
        results = []  # Clear results for the next session

        # Increment session number
        session_number += 1

        # Display break screen
        remaining_trials = TOTAL_TRIALS - (trial_index + 1)
        screen.fill(WHITE)
        rest_lines = [BREAK_TEXT, f"Trials Remaining: {remaining_trials}"]
        for i, line in enumerate(rest_lines):
            text_surface = font.render(line, True, BLACK)
            text_rect = text_surface.get_rect(center=(WINDOW_WIDTH // 2, 200 + i * 50))
            screen.blit(text_surface, text_rect)
        pygame.display.flip()
```

```
        waiting_for_space = True
        while waiting_for_space:
            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    pygame.quit()
                    quit()
                elif event.type == pygame.KEYDOWN and event.key == pygame.K_SPACE:
                    waiting_for_space = False

# Save final session results
if results:
    save_results(participant_name, participant_number, results, is_first_write=first_write)

# End of experiment
screen.fill(WHITE)
text_surface = font.render("Experiment Completed! Results saved.", True, BLACK)
text_rect = text_surface.get_rect(center=(WINDOW_WIDTH // 2, WINDOW_HEIGHT // 2))
screen.blit(text_surface, text_rect)
pygame.display.flip()
pygame.time.delay(3000)
```

```
pygame.quit()
```

Experiment 2:

```
import pygame

import os

import random

import csv

import time


# ==============================

# Customizable Variables

# ==============================


# Window settings

WINDOW_WIDTH = 1200

WINDOW_HEIGHT = 800


# Time settings (in seconds)

FIXATION_TIME = 0.5          # Duration of fixation cross

DISTRACTOR_ONLY_TIME = 1.0    # Duration to display the distractor alone

RESPONSE_WINDOW = 2.0        # Time allowed for participant to respond

FEEDBACK_TIME = 0.5          # Duration of feedback screen
```

```
# Break settings
BREAK_INTERVAL = 25          # Number of trials between breaks
BREAK_TEXT = "Take a short break! Press SPACE to continue."

# Trial settings
TOTAL_TRIALS = 200           # Total number of trials in the experiment

# Instructions
INSTRUCTIONS = ("Primary Task: Categorize the shape in the center.\n"
                "Press J for Circle, K for Square, L for Triangle.\n"
                "Ignore the image around the shape.\n"
                "Press SPACE to start.")

# Paths to stimuli
SHAPES_PATH = {
    "circle": "shapes/circle",
    "square": "shapes/square",
    "triangle": "shapes/triangle"
}

DISTRACTORS_PATHS = {
    "happy": "distractors/happy",
```

```
    "angry": "distractors/angry",
    "neutral": "distractors/neutral"
}

# Key mappings for the primary task
key_mapping = {"circle": pygame.K_j, "square": pygame.K_k, "triangle": pygame.K_l}

# ==============================
# Experiment Code
# ==============================

pygame.init()

# Screen settings
screen = pygame.display.set_mode((WINDOW_WIDTH, WINDOW_HEIGHT))
pygame.display.set_caption("Emotion Categorization Experiment 2")
font = pygame.font.Font(None, 50)

WHITE = (255, 255, 255)
BLACK = (0, 0, 0)

# Function to display participant info input form
```

```
def get_participant_info():
    participant_name = ""
    participant_number = ""
    active_field = "name"

    instructions = [
        "Enter Participant Name:",
        "Enter Participant Number:",
        "Press ENTER to start the experiment."
    ]

    while True:
        screen.fill(WHITE)

        for i, text in enumerate(instructions):
            text_surface = font.render(text, True, BLACK)
            text_rect = text_surface.get_rect(center=(WINDOW_WIDTH // 2, 100 + i * 100))
            screen.blit(text_surface, text_rect)

        name_surface = font.render(f"Name: {participant_name}", True, BLACK)
        name_rect = name_surface.get_rect(center=(WINDOW_WIDTH // 2, 400))
        screen.blit(name_surface, name_rect)
```

```
number_surface = font.render(f"Number: {participant_number}", True, BLACK)
number_rect = number_surface.get_rect(center=(WINDOW_WIDTH // 2, 500))
screen.blit(number_surface, number_rect)

pygame.display.flip()

for event in pygame.event.get():
    if event.type == pygame.QUIT:
        pygame.quit()
        return None, None

    if event.type == pygame.KEYDOWN:
        if active_field == "name":
            if event.key == pygame.K_RETURN:
                active_field = "number"
            elif event.key == pygame.K_BACKSPACE:
                participant_name = participant_name[:-1]
            else:
                participant_name += event.unicode
        elif active_field == "number":
            if event.key == pygame.K_RETURN:
```

```
                return participant_name.strip(), participant_number.strip()
            elif event.key == pygame.K_BACKSPACE:
                participant_number = participant_number[:-1]
            else:
                participant_number += event.unicode

# Function to display experiment instructions
def display_instructions():
    screen.fill(WHITE)
    lines = INSTRUCTIONS.split("\n")
    for i, line in enumerate(lines):
        text_surface = font.render(line, True, BLACK)
        text_rect = text_surface.get_rect(center=(WINDOW_WIDTH // 2, 200 + i * 50))
        screen.blit(text_surface, text_rect)
    pygame.display.flip()

    waiting_for_space = True
    while waiting_for_space:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                quit()
```

```
            elif event.type == pygame.KEYDOWN and event.key == pygame.K_SPACE:
                waiting_for_space = False


# Save results to CSV
def save_results_to_csv(filename, results, is_first_write=False):
    with open(filename, mode="a", newline="") as file:
        fieldnames = [
            "session_number",
            "distractor_type",
            "shape",
            "response",
            "reaction_time",
            "correctness"
        ]
        writer = csv.DictWriter(file, fieldnames=fieldnames)
        if is_first_write:
            writer.writeheader()  # Write header only once
        writer.writerows(results)


# Load all images from subfolders
def load_images(folder):
    if not os.path.exists(folder):
```

```
        raise FileNotFoundError(f"Folder not found: {folder}")
    images = [os.path.join(folder, f) for f in os.listdir(folder) if f.lower().endswith((".jpg", ".jpeg", ".png"))]
    print(f"Debug: Found {len(images)} images in {folder}")  # Debugging line
    if not images:
        raise ValueError(f"No valid images found in folder: {folder}")
    return images

# Load distractors
def load_distractors():
    distractors = {"happy": [], "angry": [], "neutral": []}
    for category, folder in DISTRACTORS_PATHS.items():
        distractors[category].extend(load_images(folder))
    return distractors

# Load shapes
def load_shapes():
    shapes = {"circle": [], "square": [], "triangle": []}
    for shape, folder in SHAPES_PATH.items():
        shapes[shape].extend(load_images(folder))
    return shapes

# Trial generation
```

```
def generate_trials():
    trials = []
    shapes = list(SHAPES_PATH.keys())
    distractor_types = list(DISTRACTORS_PATHS.keys())

    for _ in range(TOTAL_TRIALS):
        shape = random.choice(shapes)
        distractor_type = random.choice(distractor_types)
        trials.append({"shape": shape, "distractor_type": distractor_type})
    random.shuffle(trials)
    return trials

# Run experiment
def run_experiment(trials, shapes, distractors, output_file):
    results = []
    session_number = 1
    is_first_write = True

    for trial_index, trial in enumerate(trials):
        # Fixation cross
        screen.fill(WHITE)
        fixation = font.render("+", True, BLACK)
```

```
        fixation_rect = fixation.get_rect(center=(WINDOW_WIDTH // 2, WINDOW_HEIGHT // 2))
        screen.blit(fixation, fixation_rect)
        pygame.display.flip()
        pygame.time.delay(int(FIXATION_TIME * 1000))

        # Load distractor and shape
        distractor_img_path = random.choice(distractors[trial["distractor_type"]])
        distractor_img = pygame.image.load(distractor_img_path)
        distractor_img = pygame.transform.scale(distractor_img, (400, 400))
        distractor_rect = distractor_img.get_rect(center=(WINDOW_WIDTH // 2,
WINDOW_HEIGHT // 2))

        shape_img_path = random.choice(shapes[trial["shape"]])
        shape_img = pygame.image.load(shape_img_path)
        shape_img = pygame.transform.scale(shape_img, (200, 200))
        shape_rect = shape_img.get_rect(center=(WINDOW_WIDTH // 2, WINDOW_HEIGHT //
2))

        # Display distractor only (for 1 second)
        screen.fill(WHITE)
        screen.blit(distractor_img, distractor_rect)
        pygame.display.flip()
```

```
        pygame.time.delay(int(DISTRACTOR_ONLY_TIME * 1000))  # Display distractor for 1 second


        # Display distractor + shape
        screen.fill(WHITE)
        screen.blit(distractor_img, distractor_rect)  # Draw distractor first
        screen.blit(shape_img, shape_rect)          # Overlay shape
        pygame.display.flip()


        # Collect response
        start_time = time.time()
        response = None
        reaction_time = None
        correct = False


        while time.time() - start_time < RESPONSE_WINDOW:
            for event in pygame.event.get():
                if event.type == pygame.KEYDOWN:
                    response = event.key
                    reaction_time = time.time() - start_time
                    correct = key_mapping[trial["shape"]] == response
                    break
            if response:
```

```
            break


    # Log trial result
    response_str = next((key for key, value in key_mapping.items() if value == response), "No Response")
    results.append({
        "session_number": session_number,
        "distractor_type": trial["distractor_type"],
        "shape": trial["shape"],
        "response": response_str,
        "reaction_time": reaction_time if response else "No Response",
        "correctness": "Correct" if correct else "Incorrect"
    })


    # Feedback
    feedback_color = (0, 255, 0) if correct else (255, 0, 0)
    feedback_text = "Correct" if correct else "Incorrect"
    screen.fill(WHITE)
    feedback_surface = font.render(feedback_text, True, feedback_color)
    feedback_rect = feedback_surface.get_rect(center=(WINDOW_WIDTH // 2, WINDOW_HEIGHT // 2))
    screen.blit(feedback_surface, feedback_rect)
    pygame.display.flip()
```

```
pygame.time.delay(int(FEEDBACK_TIME * 1000))


# Break after specified interval
if (trial_index + 1) % BREAK_INTERVAL == 0 or trial_index + 1 == len(trials):
    save_results_to_csv(output_file, results, is_first_write)
    is_first_write = False
    results = []  # Clear results for the next session
    session_number += 1


    # Display break screen
    remaining_trials = TOTAL_TRIALS - (trial_index + 1)
    screen.fill(WHITE)
    rest_lines = [BREAK_TEXT, f"Trials Remaining: {remaining_trials}"]
    for i, line in enumerate(rest_lines):
        text_surface = font.render(line, True, BLACK)
        text_rect = text_surface.get_rect(center=(WINDOW_WIDTH // 2, 200 + i * 50))
        screen.blit(text_surface, text_rect)
    pygame.display.flip()


    waiting_for_space = True
    while waiting_for_space:
        for event in pygame.event.get():
```

```
            if event.type == pygame.QUIT:
                pygame.quit()
                quit()
            elif event.type == pygame.KEYDOWN and event.key == pygame.K_SPACE:
                waiting_for_space = False

# Main execution
def main():
    participant_name, participant_number = get_participant_info()
    if not participant_name or not participant_number:
        pygame.quit()
        return

    output_file = f"{participant_name}_{participant_number}_exp2_results.csv"

    display_instructions()
    distractors = load_distractors()
    shapes = load_shapes()
    trials = generate_trials()
    run_experiment(trials, shapes, distractors, output_file)
    print(f"Experiment completed. Results saved to {output_file}")
```

```
if __name__ == "__main__":
    main()
    pygame.quit()
```