ITADAKI - Recipe Image Retrieval (RAW)

EfficientNet (gelé) pour retrouver les top-3 recettes similaires à partir d'une image

Objectif:

- Analyse exploratoire du dataset
- Tester l'extraction d'embedding et la recherche par similarité avec EfficientNet gelé

---

1. Imports et config

In [1]:
```
import tensorflow as tf
from tensorflow.keras.applications import EfficientNetB0
from tensorflow.keras.layers import *
from tensorflow.keras.models import Model

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import cv2
import os
import random
from PIL import Image
import re
import warnings
warnings.filterwarnings('ignore')

# Dark theme
plt.style.use('dark_background')
sns.set_palette("husl")
```

```python
# Seed pour reproductibilité
SEED = 42
random.seed(SEED)
np.random.seed(SEED)
tf.random.set_seed(SEED)

# HYPERPARAMETRES - Recipe Image Retrieval avec EfficientNet
CONFIG = {
    # === IMAGE PARAMETERS ===
    'IMG_SIZE': 224,                    # Optimal pour EfficientNet
    'BATCH_SIZE': 32,                   # Bon équilibre mémoire/performance

    # === EMBEDDING PARAMETERS ===
    'EMBEDDING_DIM_NATIVE': 1280,       #  EfficientNet features natives
    'EMBEDDING_DIM_CUSTOM': 512,        #  Custom head output (si fine-tuning)

    # === PHASE 1: FEATURE EXTRACTION ===
    'USE_NATIVE_FEATURES': True,        #  True = 1280-dim, False = 512-dim custom

    # === PHASE 2: TRANSFER LEARNING ===
    'TRANSFER_EPOCHS': 10,              #  Entraînement head seulement
    'TRANSFER_LR': 0.001,               #  Learning rate plus élevé pour nouvelles couches
    'TRANSFER_BATCH_SIZE': 16,          #  Plus petit pour stabilité

    # === PHASE 3: FINE-TUNING ===
    'FINETUNE_EPOCHS': 15,              #  Fine-tuning complet
    'FINETUNE_LR': 0.0001,              #  Learning rate faible pour pré-entraîné
    'FINETUNE_UNFREEZE_LAYERS': 20,     #  Nombre de couches à dégeler

    # === TRAINING PARAMETERS ===
    'TRIPLET_MARGIN': 0.3,              #  Augmenté pour meilleure séparation
    'DROPOUT_RATE': 0.3,                #  Régularisation
    'WEIGHT_DECAY': 0.0001,             #  L2 regularization

    # === OPTIMIZATION ===
    'OPTIMIZER': 'adam',                #  Standard
    'PATIENCE': 5,                      #  Early stopping
    'REDUCE_LR_PATIENCE': 3,            #  Réduction LR
    'REDUCE_LR_FACTOR': 0.5,            #  Facteur réduction
```

```
    # === DATA AUGMENTATION ===
    'USE_AUGMENTATION': True,           #  Pour fine-tuning
    'ROTATION_RANGE': 15,               #  Rotation aléatoire
    'ZOOM_RANGE': 0.1,                  #  Zoom aléatoire
    'HORIZONTAL_FLIP': True,            #  Flip horizontal

    # === EVALUATION ===
    'SIMILARITY_THRESHOLD': 0.7,       #  Seuil similarité
    'TOP_K_RESULTS': 3,                 #  Nombre résultats retournés
    'VALIDATION_SPLIT': 0.2,           #  Split pour validation
}

#  CONFIGS SPÉCIFIQUES PAR PHASE
CONFIG_PHASE1 = {**CONFIG, **{
    'MODE': 'feature_extraction',
    'EMBEDDING_DIM': CONFIG['EMBEDDING_DIM_NATIVE'],
    'TRAINABLE': False
}}

print("Configuration adaptée pour EfficientNet")
print(f"Phase 1: extraction d'embedding simple ({CONFIG['EMBEDDING_DIM_NATIVE']}-dim)")

print("Imports et configuration OK!")
print(f"TensorFlow: {tf.__version__}")
print(f"GPU: {tf.config.list_physical_devices('GPU')}")
```

```
Configuration adaptée pour EfficientNet
Phase 1: extraction d'embedding simple (1280-dim)
Imports et configuration OK!
TensorFlow: 2.19.0
GPU: []
```

2. Chargement du dataset

In [2]:

```
try:
    import kagglehub
    dataset_path = kagglehub.dataset_download("pes12017000148/food-ingredients-and-recipe-dataset-with-images")
    print(f"Dataset téléchargé: {dataset_path}")

    images_dir = os.path.join(dataset_path, "Food Images", "Food Images")
```

```
    csv_files = [f for f in os.listdir(dataset_path) if f.endswith('.csv')]
    csv_path = os.path.join(dataset_path, csv_files[0]) if csv_files else None

    print(f"Images: {images_dir}")
    print(f"CSV: {csv_path}")

except ImportError:
    print("kagglehub non installé, veuillez l'installer avec: pip install kagglehub")
    exit()

# Charger le CSV
df = pd.read_csv(csv_path)
print(f"{len(df)} recettes chargées")
print(f"Colonnes: {list(df.columns)}")

# Nettoyage basique
df_clean = df.dropna(subset=['Title', 'Ingredients', 'Instructions', 'Image_Name']).copy()
print(f"Après nettoyage: {len(df_clean)} recettes")

# Créer le texte complet
def clean_text(text):
    if pd.isna(text):
        return ""
    text = str(text).lower().strip()
    text = re.sub(r'[^a-zA-Z0-9\s,.!?-]', '', text)
    text = re.sub(r'\s+', ' ', text)
    return text

df_clean['full_text'] = (
    df_clean['Title'].apply(clean_text) + ' ' +
    df_clean['Ingredients'].apply(clean_text) + ' ' +
    df_clean['Instructions'].apply(clean_text)
)

print("Données chargées")
```

```
Dataset téléchargé: C:\Users\Naturel\.cache\kagglehub\datasets\pes12017000148\food-ingredients-and-recipe-dataset-with-images\versions\1
Images: C:\Users\Naturel\.cache\kagglehub\datasets\pes12017000148\food-ingredients-and-recipe-dataset-with-images\versions\1\Food Images\Food Images
CSV: C:\Users\Naturel\.cache\kagglehub\datasets\pes12017000148\food-ingredients-and-recipe-dataset-with-images\versions\1\Food Ingredients and Recipe Dataset with Image Name Mapping.csv
13501 recettes chargées
Colonnes: ['Unnamed: 0', 'Title', 'Ingredients', 'Instructions', 'Image_Name', 'Cleaned_Ingredients']
Après nettoyage: 13493 recettes
Données chargées
```

In [3]:

```python
print("Vérification des images...")

if not os.path.exists(images_dir):
    print(f"Dossier non trouvé: {images_dir}")
    exit()

available_images = set(os.listdir(images_dir))
print(f"{len(available_images)} images trouvées")

print("- Exemples d'images:")
for i, img in enumerate(list(available_images)[:5]):
    print(f"{img}")

print("- Exemples de noms CSV:")
for i, name in enumerate(df_clean['Image_Name'].dropna().head(5)):
    print(f"{name}")

# Fonction de correspondance
def find_image_path(image_name):
    """Trouve le chemin complet de l'image"""
    if pd.isna(image_name):
        return None

    # Essayer nom exact
    if image_name in available_images:
        return os.path.join(images_dir, image_name)

    # Essayer avec .jpg
    if f"{image_name}.jpg" in available_images:
        return os.path.join(images_dir, f"{image_name}.jpg")
```

```
    # Essayer avec .jpeg
    if f"{image_name}.jpeg" in available_images:
        return os.path.join(images_dir, f"{image_name}.jpeg")

    # Essayer avec .png
    if f"{image_name}.png" in available_images:
        return os.path.join(images_dir, f"{image_name}.png")

    return None

df_clean['image_path'] = df_clean['Image_Name'].apply(find_image_path)
df_clean['has_image'] = df_clean['image_path'].notna()

# Filtrer les recettes avec images
recipes_with_images = df_clean[df_clean['has_image']].copy()
print(f"\n{len(recipes_with_images)} recettes avec images conservées dans recipes_with_images")
```

```
Vérification des images...
13582 images trouvées
- Exemples d'images:
arugula-and-mint-salad-with-oil-cured-black-olives-oranges-and-ricotta-salata-236186.jpg
watermelon-granita-with-gingered-strawberries-360529.jpg
turkey-tetrazzini-13377.jpg
cucumber-sake-tini-358951.jpg
chawan-mushi-with-shrimp-and-spring-peas-51155250.jpg
- Exemples de noms CSV:
miso-butter-roast-chicken-acorn-squash-panzanella
crispy-salt-and-pepper-potatoes-dan-kluger
thanksgiving-mac-and-cheese-erick-williams
italian-sausage-and-bread-stuffing-240559
newtons-law-apple-bourbon-cocktail

13463 recettes avec images conservées dans recipes_with_images
```

3. Nettoyage NLP

In [4]:
```
#  Nettoyage NLP
import nltk
import pandas as pd
```

```
import re

# NLTK
print("Téléchargement des ressources NLTK...")
nltk.download('stopwords', quiet=True)
nltk.download('punkt_tab', quiet=True)  # Version récente de punkt
nltk.download('wordnet', quiet=True)
nltk.download('omw-1.4', quiet=True)  # Pour WordNet multilingue
print("Ressources NLTK OK!")

from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
import string

# Configuration NLTK avec stop words culinaires personnalisés
stop_words = set(stopwords.words('english'))

# Stop words spécifiques à la cuisine, pourrait être amélioré
culinary_stopwords = {
    'bake',
    'baking',
    'chopped',
    'coarse',
    'coarsely',
    'cook',
    'cooking',
    'cube',
    'cubes',
    'cup',
    'cups',
    'cut',
    'diced',
    'divided',
    'finely',
    'gram',
    'grams',
    'grated',
    'halved',
    'hour',
    'hours',
```

```
    'inch',
    'inches',
    'kg',
    'lengthwise',
    'liter',
    'liters',
    'melted',
    'minute',
    'minutes',
    'minced',
    'ml',
    'oz',
    'peeled',
    'piece',
    'pieces',
    'plus',
    'pound',
    'pounds',
    'quartered',
    'recipe',
    'recipes',
    'room',
    'seeded',
    'serve',
    'serves',
    'serving',
    'servings',
    'sliced',
    'stick',
    'tablespoon',
    'tablespoons',
    'tbsp',
    'teaspoon',
    'teaspoons',
    'temperature',
    'thinly',
    'tsp',
    'trimmed'
}

# Fusionner les stop words
```

```
stop_words.update(culinary_stopwords)
print(f"Stop words enrichis: {len(stop_words)} mots au total comprenant {len(culinary_stopwords)} stop words culinaires")

lemmatizer = WordNetLemmatizer()

def clean_text_advanced(text):
    """
    Nettoyage avancé du texte avec NLTK
    """
    if pd.isna(text) or text == '':
        return ''

    # Convertir en string et minuscules
    text = str(text).lower()

    # Supprimer la ponctuation et caractères spéciaux
    text = text.translate(str.maketrans('', '', string.punctuation))

    # Supprimer les chiffres
    text = re.sub(r'\d+', '', text)

    # Tokenisation
    tokens = word_tokenize(text)

    # Supprimer stopwords et mots courts (< 3 caractères)
    tokens = [token for token in tokens if token not in stop_words and len(token) > 2]

    # Lemmatisation
    tokens = [lemmatizer.lemmatize(token) for token in tokens]

    return ' '.join(tokens)

def extract_ingredients_list(ingredients_str):
    """
    Extrait et nettoie la liste d'ingrédients depuis le format string
    """
    if pd.isna(ingredients_str):
        return []

    # Supprimer les crochets et guillemets
    cleaned = str(ingredients_str).strip("[]'\"")
```

```
    # Séparer par virgules et nettoyer
    ingredients = []
    for item in cleaned.split(','):
        item = item.strip("' \"")
        if item and len(item) > 1:
            # Nettoyer chaque ingrédient
            clean_ingredient = clean_text_advanced(item)
            if clean_ingredient:
                ingredients.append(clean_ingredient)

    return ingredients

# Nettoyage NLP
print("Application du nettoyage NLP...")
recipes_with_images['Title_Clean'] = recipes_with_images['Title'].apply(clean_text_advanced)
recipes_with_images['Ingredients_Clean'] = recipes_with_images['Ingredients'].apply(clean_text_advanced)
recipes_with_images['Instructions_Clean'] = recipes_with_images['Instructions'].apply(clean_text_advanced)

# Extraire la liste des ingrédients
recipes_with_images['Ingredients_List'] = recipes_with_images['Ingredients'].apply(extract_ingredients_list)

# Recréer le texte complet avec le nettoyage avancé
recipes_with_images['full_text_advanced'] = (
    recipes_with_images['Title_Clean'] + ' ' +
    recipes_with_images['Ingredients_Clean'] + ' ' +
    recipes_with_images['Instructions_Clean']
)

print("Nettoyage NLP terminé!")
```

```
Téléchargement des ressources NLTK...
Ressources NLTK OK!
Stop words enrichis: 257 mots au total comprenant 59 stop words culinaires
Application du nettoyage NLP...
Nettoyage NLP terminé!
```

4. Visualisation des images du dataset

In [5]:
```python
def visualize_recipe_gallery_harmonized(recipes_df, images_dir, grid_size=(4, 3), random_seed=42):
    """
    Affiche une galerie d'images de recettes
    """
    # Configuration du style
    plt.style.use('dark_background')

    # Créer la figure avec le fond sombre harmonisé
    fig, axes = plt.subplots(grid_size[0], grid_size[1], figsize=(18, 16))
    fig.patch.set_facecolor('#1a1a1a')  # Même fond que les autres visualisations

    # Titre principal avec style harmonisé
    fig.suptitle('GALERIE DE RECETTES',
                 fontsize=24, fontweight='bold', color='white', y=0.98)

    # Aplatir les axes pour faciliter l'itération
    axes = axes.flatten()

    # Échantillonner des recettes aléatoirement
    total_samples = grid_size[0] * grid_size[1]
    if len(recipes_df) >= total_samples:
        sample_recipes = recipes_df.sample(n=total_samples, random_state=random_seed)
    else:
        sample_recipes = recipes_df.sample(n=len(recipes_df), random_state=random_seed)
        print(f"Seulement {len(recipes_df)} recettes disponibles (demandé: {total_samples})")

    # Palette de couleurs pour les bordures
    border_colors = ['#FF6B9D', '#4ECDC4', '#45B7D1', '#96CEB4', '#FECA57',
                     '#FF9FF3', '#54A0FF', '#5F27CD', '#A8E6CF', '#FFD93D']

    for idx, (_, recipe) in enumerate(sample_recipes.iterrows()):
        if idx >= len(axes):
            break

        ax = axes[idx]
        ax.set_facecolor('#2d2d2d')

        try:
            # Utiliser le chemin d'image déjà résolu
            image_path = recipe.get('image_path')
```

```
        if image_path and os.path.exists(image_path):
            # Charger et afficher l'image
            img = Image.open(image_path)
            img = img.convert('RGB')
            ax.imshow(img)

            # Titre tronqué si trop long
            title = recipe['Title']
            if len(title) > 35:
                title = title[:35] + "..."

            border_color = border_colors[idx % len(border_colors)]

            ax.set_title(title, fontsize=11, fontweight='bold',
                         color='white', pad=15,
                         bbox=dict(boxstyle="round,pad=0.5",
                                   facecolor='#1a1a1a',
                                   edgecolor=border_color,
                                   linewidth=2, alpha=0.9))
            ax.axis('off')

            # Ajouter une bordure colorée autour de l'image
            for spine in ax.spines.values():
                spine.set_visible(True)
                spine.set_color(border_color)
                spine.set_linewidth(3)
                spine.set_alpha(0.8)

        else:
            # Placeholder pour images manquantes
            ax.text(0.5, 0.5, f'\n\nImage non trouvée\n\n"{recipe["Title"][:25]}..."',
                    horizontalalignment='center', verticalalignment='center',
                    transform=ax.transAxes, fontsize=10, color='white',
                    bbox=dict(boxstyle="round,pad=0.8",
                              facecolor='#2d2d2d',
                              edgecolor='#FF6B9D',
                              linewidth=2, alpha=0.9))
            ax.set_title("Image manquante", fontsize=11, color='#FF6B9D', fontweight='bold')
            ax.axis('off')
```

```
        except Exception as e:
            ax.text(0.5, 0.5, f'\n\nErreur de chargement\n\n{str(e)[:20]}...',
                   horizontalalignment='center', verticalalignment='center',
                   transform=ax.transAxes, fontsize=10, color='white',
                   bbox=dict(boxstyle="round,pad=0.8",
                           facecolor='#2d2d2d',
                           edgecolor='#FECA57',
                           linewidth=2, alpha=0.9))
            ax.set_title("Erreur", fontsize=11, color='#FECA57', fontweight='bold')
            ax.axis('off')

    # Masquer des axes non utilisés
    for idx in range(len(sample_recipes), len(axes)):
        axes[idx].set_facecolor('#1a1a1a')
        axes[idx].axis('off')

    plt.tight_layout(pad=3.0, rect=[0, 0.03, 1, 0.96])
    plt.show()

    # Statistiques
    print(f"\nSTATISTIQUES SUR LES IMAGES:")
    print(f"Images affichées: {min(len(sample_recipes), len(axes))}")
    print(f"Total recettes disponibles: {len(recipes_df):,}")
    print(f"Recettes avec images: {recipes_df['has_image'].sum():,}")
    print(f"Pourcentage avec images: {(recipes_df['has_image'].sum() / len(recipes_df) * 100):.1f}%")

    # Afficher quelques exemples de titres
    print(f"\nEXEMPLES DE TITRES:")
    for i, (_, recipe) in enumerate(sample_recipes.head(3).iterrows()):
        print(f"{i+1}. {recipe['Title']}")

#  Affichage de la galerie
if 'recipes_with_images' in globals() and 'images_dir' in globals():
    if len(recipes_with_images) > 0:
        visualize_recipe_gallery_harmonized(recipes_with_images, images_dir, grid_size=(4, 3))
    else:
        print("Aucune recette avec image trouvée pour la galerie")

else:
    print("Erreur. Assurez-vous d'avoir exécuté les cellules de chargement des données.")
    if 'recipes_with_images' not in globals():
```

```
        print("- 'recipes_with_images' non trouvé")
    if 'images_dir' not in globals():
        print("- 'images_dir' non trouvé")
```

GALERIE DE RECETTES

Smoked Salmon, Fennel and Goat Chee...

Spicy Thai Tofu with Red Bell Peppe...

Steak Salad with Caraway Vinaigrett...

Apple Pandowdy

Tex-Mex Mole

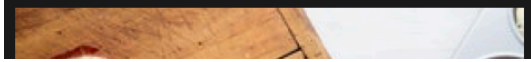
Perfect Coddled Egg

Squid Ink Pasta with Shrimp, Nduja,...

Harvest Pear Crisp with Candied Gin...

Vanilla Crumb

Eggs Baked in Crispy Prosciutto Bas...



Oatmeal Cookie Energy Bites

Pork Tenderloin Stir-Fry with Tange...

```
STATISTIQUES SUR LES IMAGES:
Images affichées: 12
Total recettes disponibles: 13,463
Recettes avec images: 13,463
Pourcentage avec images: 100.0%

EXEMPLES DE TITRES:
1. Smoked Salmon, Fennel and Goat Cheese Toasts
2. Spicy Thai Tofu with Red Bell Peppers and Peanuts
3. Steak Salad with Caraway Vinaigrette and Rye Croutons
```

5. Analyse exploratoire des données (recipes_with_images)

In [6]:
```python
import platform
from wordcloud import WordCloud

# Configuration  dark theme
plt.style.use('dark_background')
plt.rcParams['font.family'] = ['Segoe UI Emoji', 'Apple Color Emoji', 'Noto Color Emoji', 'DejaVu Sans', 'sans-serif']
plt.rcParams['axes.unicode_minus'] = False

if platform.system() == "Windows":
    plt.rcParams['font.family'] = ['Segoe UI Emoji', 'DejaVu Sans', 'sans-serif']
elif platform.system() == "Linux":
    plt.rcParams['font.family'] = ['Noto Color Emoji', 'DejaVu Sans', 'Liberation Sans', 'sans-serif']
elif platform.system() == "Darwin":  # macOS
    plt.rcParams['font.family'] = ['Apple Color Emoji', 'SF Pro Display', 'DejaVu Sans', 'sans-serif']
else:
    plt.rcParams['font.family'] = ['DejaVu Sans', 'sans-serif']

plt.rcParams['axes.unicode_minus'] = False
```

```
warnings.filterwarnings('ignore', category=UserWarning, module='matplotlib.font_manager')

# Palette de couleurs magnifique (IDENTIQUE)
beautiful_colors = ['#FF6B9D', '#4ECDC4', '#45B7D1', '#96CEB4', '#FECA57', '#FF9FF3', '#54A0FF', '#5F27CD', '#A8E6CF', '#FFD93

def create_wordcloud_harmonized(text_data, title, max_words=100, colormap='viridis'):
    """
    Crée un nuage de mots
    """
    # Combiner tout le texte
    if isinstance(text_data, list):
        combined_text = ' '.join([str(text) for text in text_data if pd.notna(text) and str(text).strip()])
    else:
        combined_text = ' '.join([str(text) for text in text_data.dropna() if str(text).strip()])

    if not combined_text.strip():
        print(f" Pas de données texte pour {title}")
        return None

    # Créer le WordCloud avec le style harmonisé
    wordcloud = WordCloud(
        width=800,
        height=400,
        background_color='#2d2d2d',  # Même fond que les autres graphiques
        max_words=max_words,
        colormap=colormap,
        collocations=False,
        prefer_horizontal=0.7
    ).generate(combined_text)

    return wordcloud

def plot_top_ingredients_harmonized(recipes_df, ax, top_n=20):
    """
    Graphique des ingrédients
    """
    ax.set_facecolor('#2d2d2d')  # Même fond

    # Compter tous les ingrédients
    all_ingredients = []
```

```
    if 'Ingredients_List' in recipes_df.columns:
        for ingredients_list in recipes_df['Ingredients_List'].dropna():
            if isinstance(ingredients_list, list):
                all_ingredients.extend(ingredients_list)
    else:
        # Fallback: extraire depuis la colonne Ingredients
        for ingredients_text in recipes_df['Ingredients'].dropna():
            if pd.notna(ingredients_text) and str(ingredients_text).strip():
                ingredients = [ing.strip().lower() for ing in str(ingredients_text).split(',')]
                all_ingredients.extend([ing for ing in ingredients if len(ing) > 2])

    if not all_ingredients:
        ax.text(0.5, 0.5, " Aucun ingrédient trouvé", ha='center', va='center',
                transform=ax.transAxes, color='white', fontsize=12)
        return None

    # Compter les fréquences
    from collections import Counter
    ingredient_counts = Counter(all_ingredients)
    top_ingredients = ingredient_counts.most_common(top_n)

    if not top_ingredients:
        return None

    # Créer le graphique avec le style harmonisé
    ingredients, counts = zip(*top_ingredients)

    # Utiliser la palette magnifique
    colors = [beautiful_colors[i % len(beautiful_colors)] for i in range(len(ingredients))]

    bars = ax.barh(range(len(ingredients)), counts,
                   color=colors, alpha=0.9,
                   edgecolor='white', linewidth=0.8)  # Bordures plus fines

    ax.set_xlabel('Fréquence', fontweight='bold', color='white', fontsize=12)
    ax.set_ylabel('Ingrédients', fontweight='bold', color='white', fontsize=12)
    ax.set_title(f' Top {top_n} Ingrédients', fontweight='bold', fontsize=16, color='white', pad=20)
    ax.set_yticks(range(len(ingredients)))
    ax.set_yticklabels(ingredients, fontsize=10, color='white')
    ax.tick_params(colors='white')
```

```
    # Ajouter les valeurs avec style
    for bar, count in zip(bars, counts):
        ax.text(bar.get_width() + max(counts)*0.02, bar.get_y() + bar.get_height()/2,
                str(count), ha='left', va='center', fontweight='bold',
                color='white', fontsize=11)

    # Style harmonisé
    ax.invert_yaxis()
    ax.grid(True, alpha=0.3, color='#404040', linestyle='--')
    ax.spines['bottom'].set_color('#404040')
    ax.spines['top'].set_color('#404040')
    ax.spines['right'].set_color('#404040')
    ax.spines['left'].set_color('#404040')

    return top_ingredients

def plot_instruction_lengths_harmonized(recipes_df, ax):
    """
    Distribution des longueurs des instructions
    """
    ax.set_facecolor('#2d2d2d')

    instruction_lengths = recipes_df['Instructions'].dropna().apply(lambda x: len(str(x).split()))

    # Histogramme
    n, bins, patches = ax.hist(instruction_lengths, bins=30, alpha=0.9,
                               edgecolor='white', linewidth=0.5)

    base_color = np.array([69, 183, 209])
    for i, patch in enumerate(patches):
        intensity = 0.4 + 0.6 * (i / len(patches))
        color = base_color * intensity / 255.0
        patch.set_facecolor(color)

    ax.set_xlabel('Nombre de mots dans les instructions', fontweight='bold', color='white', fontsize=12)
    ax.set_ylabel('Fréquence', fontweight='bold', color='white', fontsize=12)
    ax.set_title('Distribution des Instructions', fontweight='bold', fontsize=16, color='white', pad=20)
    ax.tick_params(colors='white')

    # Statistiques avec couleurs contrastantes pour la lisibilité
    mean_length = instruction_lengths.mean()
```

```
    median_length = instruction_lengths.median()
    ax.axvline(mean_length, color='#FFFFFF', linestyle='--', linewidth=3, alpha=0.9, label=f' Moyenne: {mean_length:.1f}')
    ax.axvline(median_length, color='#FFD700', linestyle='--', linewidth=3, alpha=0.9, label=f' Médiane: {median_length:.1f}')

    # Légende harmonisée
    legend = ax.legend(framealpha=0.9, facecolor='#2d2d2d', edgecolor='white')
    for text in legend.get_texts():
        text.set_color('white')

    ax.grid(True, alpha=0.3, color='#404040', linestyle='--')
    ax.spines['bottom'].set_color('#404040')
    ax.spines['top'].set_color('#404040')
    ax.spines['right'].set_color('#404040')
    ax.spines['left'].set_color('#404040')

    return instruction_lengths

if 'recipes_with_images' in globals():

    fig = plt.figure(figsize=(20, 14))
    fig.patch.set_facecolor('#1a1a1a')

    fig.suptitle('ANALYSE DU DATASET AVEC IMAGES',
                 fontsize=24, fontweight='bold', color='white', y=0.98)

    # WordCloud des ingrédients
    ax1 = plt.subplot(2, 2, 1)
    ax1.set_facecolor('#2d2d2d')

    if 'Ingredients_Clean' in recipes_with_images.columns:
        ingredients_text = recipes_with_images['Ingredients_Clean']
    else:
        ingredients_text = recipes_with_images['Ingredients']

    wordcloud_ingredients = create_wordcloud_harmonized(ingredients_text, "Ingrédients", max_words=120, colormap='plasma')
    if wordcloud_ingredients:
        ax1.imshow(wordcloud_ingredients, interpolation='bilinear')
        ax1.set_title('Nuage de mots - Ingrédients', fontweight='bold', fontsize=16, color='#FF6B9D', pad=20)
        ax1.axis('off')
    else:
        ax1.text(0.5, 0.5, "Pas de données d'ingrédients", ha='center', va='center',
```

```
            transform=ax1.transAxes, color='white', fontsize=14)

# WordCloud des titres
ax2 = plt.subplot(2, 2, 2)
ax2.set_facecolor('#2d2d2d')

if 'Title_Clean' in recipes_with_images.columns:
    titles_text = recipes_with_images['Title_Clean']
else:
    titles_text = recipes_with_images['Title']

wordcloud_titles = create_wordcloud_harmonized(titles_text, "Titres", max_words=120, colormap='viridis')
if wordcloud_titles:
    ax2.imshow(wordcloud_titles, interpolation='bilinear')
    ax2.set_title('Nuage de mots - Titres de Recettes', fontweight='bold', fontsize=16, color='#4ECDC4', pad=20)
    ax2.axis('off')
else:
    ax2.text(0.5, 0.5, "Pas de données de titres", ha='center', va='center',
            transform=ax2.transAxes, color='white', fontsize=14)

# Top 20 ingrédients
ax3 = plt.subplot(2, 2, 3)
print("Analyse des top 20 ingrédients...")
top_ingredients = plot_top_ingredients_harmonized(recipes_with_images, ax3, top_n=20)
if top_ingredients:
    print(f"Top 20 ingrédients: {[ing for ing, count in top_ingredients[:20]]}")

# Distribution des instructions
ax4 = plt.subplot(2, 2, 4)

instruction_lengths = plot_instruction_lengths_harmonized(recipes_with_images, ax4)

plt.tight_layout(pad=4.0, rect=[0, 0.03, 1, 1])
plt.show()

# Statistiques
print(f"\nSTATISTIQUES DES INSTRUCTIONS:")
print(f"Longueur moyenne: {instruction_lengths.mean():.1f} mots")
print(f"Longueur médiane: {instruction_lengths.median():.1f} mots")
print(f"Plus courte: {instruction_lengths.min()} mots")
print(f"Plus longue: {instruction_lengths.max()} mots")
```

```
print(f"\nSTATISTIQUES GÉNÉRALES:")
print(f"Recettes avec images: {len(recipes_with_images):,}")
print(f"Recettes avec ingrédients: {recipes_with_images['Ingredients'].notna().sum():,}")
print(f"Recettes avec instructions: {recipes_with_images['Instructions'].notna().sum():,}")
print(f"Recettes avec nettoyage avancé: {recipes_with_images['full_text_advanced'].notna().sum() if 'full_text_advanced' i

#  ANALYSE DES DOUBLONS
def clean_text_for_comparison(text):
    """Nettoie le texte pour la comparaison des doublons"""
    if pd.isna(text) or text is None:
        return ""
    return str(text).lower().strip().replace('\n', ' ').replace('\r', ' ')

print(f"\nANALYSE DES TITRES EN DOUBLE:")
recipes_with_images['title_clean_comparison'] = recipes_with_images['Title'].apply(clean_text_for_comparison)

# Compter les doublons de titres
title_counts = recipes_with_images['title_clean_comparison'].value_counts()
duplicate_titles = title_counts[title_counts > 1]
unique_titles = title_counts[title_counts == 1]

print(f"Titres uniques: {len(unique_titles):,}")
print(f"Titres en double: {len(duplicate_titles):,}")
print(f"Total d'occurrences dupliquées: {duplicate_titles.sum():,}")
print(f"Pourcentage de doublons: {(len(duplicate_titles) / len(title_counts) * 100):.1f}%")

#  Analyse des instructions en double
print(f"\nANALYSE DES INSTRUCTIONS EN DOUBLE:")
recipes_with_images['instructions_clean_comparison'] = recipes_with_images['Instructions'].apply(clean_text_for_comparison

# Filtrer les instructions vides
non_empty_instructions = recipes_with_images[recipes_with_images['instructions_clean_comparison'].str.len() > 10]

instruction_counts = non_empty_instructions['instructions_clean_comparison'].value_counts()
duplicate_instructions = instruction_counts[instruction_counts > 1]
unique_instructions = instruction_counts[instruction_counts == 1]

print(f"Instructions uniques: {len(unique_instructions):,}")
print(f"Instructions en double: {len(duplicate_instructions):,}")
print(f"Total d'occurrences dupliquées: {duplicate_instructions.sum():,}")
```

```
print(f"Pourcentage de doublons: {(len(duplicate_instructions) / len(instruction_counts) * 100):.1f}%")

#  Analyse des recettes complètement identiques (titre + instructions)
print(f"\nANALYSE DES RECETTES COMPLÈTEMENT IDENTIQUES:")
recipes_with_images['combined_clean'] = (
    recipes_with_images['title_clean_comparison'] + " ||| " +
    recipes_with_images['instructions_clean_comparison']
)

combined_counts = recipes_with_images['combined_clean'].value_counts()
identical_recipes = combined_counts[combined_counts > 1]

print(f"Recettes complètement uniques: {len(combined_counts[combined_counts == 1]):,}")
print(f"Recettes complètement identiques: {len(identical_recipes):,}")
print(f"Total de recettes dupliquées: {identical_recipes.sum():,}")

#  VISUALISATION DES DOUBLONS
fig_duplicates = plt.figure(figsize=(20, 16))
fig_duplicates.patch.set_facecolor('#1a1a1a')
fig_duplicates.suptitle(' ANALYSE VISUELLE DES DOUBLONS',
                        fontsize=22, fontweight='bold', color='white', y=0.98)

# Graphique en secteurs pour les recettes uniques vs dupliquées
ax_pie = plt.subplot(3, 3, 1)
ax_pie.set_facecolor('#2d2d2d')

unique_count = len(combined_counts[combined_counts == 1])
duplicate_count = identical_recipes.sum()

sizes = [unique_count, duplicate_count]
labels = [f'Uniques\n({unique_count:,})', f' Dupliquées\n({duplicate_count:,})']
colors = ['#4ECDC4', '#FF6B9D']
explode = (0.05, 0.1)

wedges, texts, autotexts = ax_pie.pie(sizes, labels=labels, colors=colors,
                                      autopct='%1.1f%%', explode=explode,
                                      shadow=True, startangle=90,
                                      textprops={'fontsize': 10, 'color': 'white', 'fontweight': 'bold'})

ax_pie.set_title('Uniques vs Dupliquées',
                 fontweight='bold', fontsize=14, color='white', pad=15)
```

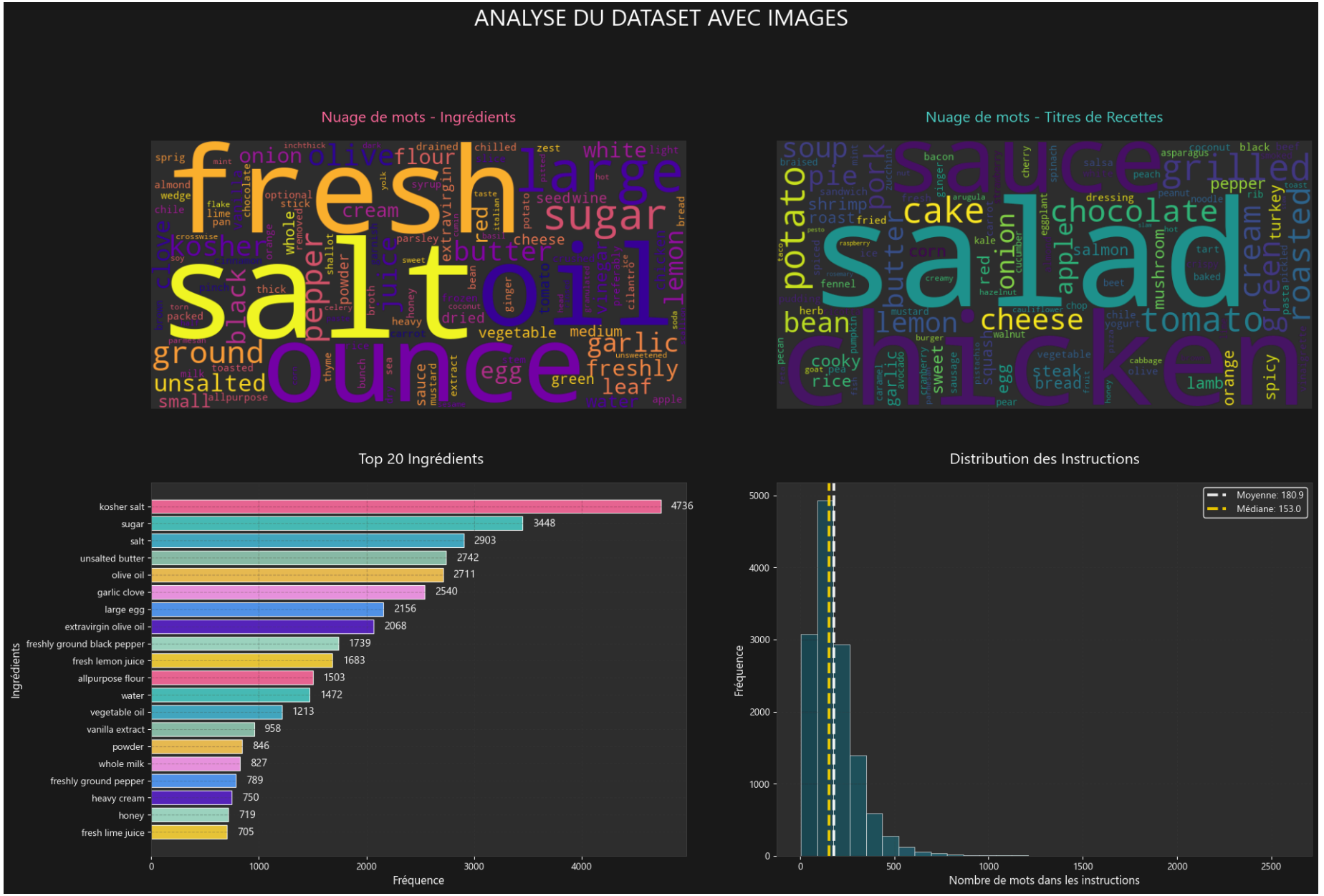
```
    # Graphique en barres des types de doublons
    ax_bar = plt.subplot(3, 3, 2)
    ax_bar.set_facecolor('#2d2d2d')

    duplicate_types = ['Titres', 'Instructions', 'Complètes']
    duplicate_counts_values = [
        len(duplicate_titles),
        len(duplicate_instructions),
        len(identical_recipes)
    ]

    bars = ax_bar.bar(duplicate_types, duplicate_counts_values,
                      color=['#FF9FF3', '#54A0FF', '#FECA57'],
                      alpha=0.9, edgecolor='white', linewidth=1.5)

    ax_bar.set_title(' Types de Doublons', fontweight='bold', fontsize=14, color='white', pad=15)
    ax_bar.set_ylabel('Nombre', fontweight='bold', color='white', fontsize=11)
    ax_bar.tick_params(colors='white', labelsize=10)

    # Ajouter les valeurs sur les barres
    for bar, count in zip(bars, duplicate_counts_values):
        ax_bar.text(bar.get_x() + bar.get_width()/2, bar.get_height() + max(duplicate_counts_values)*0.02,
                    str(count), ha='center', va='bottom', fontweight='bold',
                    color='white', fontsize=11)

    ax_bar.grid(True, alpha=0.3, color='#404040', linestyle='--')
    for spine in ax_bar.spines.values():
        spine.set_color('#404040')

    # 3. Texte d'information
    ax_info = plt.subplot(3, 3, 3)
    ax_info.set_facecolor('#2d2d2d')
    ax_info.axis('off')

    info_text = f"""STATISTIQUES DOUBLONS

Total recettes: {len(recipes_with_images):,}
Titres dupliqués: {len(duplicate_titles):,}
Instructions dupliquées: {len(duplicate_instructions):,}
Recettes identiques: {len(identical_recipes):,}
```

```
Pourcentage de doublons:
- Titres: {(len(duplicate_titles) / len(title_counts) * 100):.1f}%
- Instructions: {(len(duplicate_instructions) / len(instruction_counts) * 100):.1f}%"""

    ax_info.text(0.05, 0.95, info_text, transform=ax_info.transAxes,
                fontsize=11, color='white', va='top', ha='left',
                bbox=dict(boxstyle="round,pad=0.3", facecolor='#404040', alpha=0.8))

    # 4-7. Exemple de recette dupliquée avec images (4 sous-plots pour les images)
    if len(identical_recipes) > 0:
        # Prendre le premier doublon avec le plus d'occurrences
        most_duplicated = identical_recipes.head(1)
        duplicate_key = most_duplicated.index[0]
        duplicate_count_example = most_duplicated.iloc[0]

        # Trouver toutes les recettes identiques
        duplicate_recipes = recipes_with_images[recipes_with_images['combined_clean'] == duplicate_key]

        # Titre de l'exemple
        title_part = duplicate_key.split(" ||| ")[0]

        # Informations de l'exemple
        ax_example_info = plt.subplot(3, 1, 2)
        ax_example_info.set_facecolor('#2d2d2d')
        ax_example_info.axis('off')

        ax_example_info.text(0.5, 0.9, f'EXEMPLE DE RECETTE DUPLIQUÉE ({duplicate_count_example} fois)',
                             ha='center', va='top', transform=ax_example_info.transAxes,
                             fontsize=16, fontweight='bold', color='#FFD93D')

        ax_example_info.text(0.5, 0.7, f'Titre: "{title_part}"',
                             ha='center', va='top', transform=ax_example_info.transAxes,
                             fontsize=12, fontweight='bold', color='white')

        # Instructions preview
        instructions_part = duplicate_key.split(" ||| ")[1]
        preview_instructions = instructions_part[:150] + "..." if len(instructions_part) > 150 else instructions_part
        ax_example_info.text(0.5, 0.4, f'Instructions: "{preview_instructions}"',
                             ha='center', va='top', transform=ax_example_info.transAxes,
                             fontsize=10, color='#A8E6CF', style='italic')
```

```
    # Images des recettes dupliquées (4 subplots)
    valid_images = []
    for idx, recipe in duplicate_recipes.head(4).iterrows():
        if 'image_path' in recipe and pd.notna(recipe['image_path']):
            try:
                img_path = recipe['image_path']
                if os.path.exists(img_path):
                    valid_images.append((img_path, recipe['Title']))
            except:
                pass

    # Positions pour les 4 images en bas
    image_positions = [(3, 4, 9), (3, 4, 10), (3, 4, 11), (3, 4, 12)]  # 3 rows, 4 cols, positions 9-12

    for i in range(4):
        ax_img = plt.subplot(*image_positions[i])
        ax_img.set_facecolor('#2d2d2d')

        if i < len(valid_images):
            try:
                img_path, original_title = valid_images[i]
                img = plt.imread(img_path)
                ax_img.imshow(img)
                ax_img.set_title(f'Copie #{i+1}', fontsize=11, color='white', fontweight='bold', pad=10)

                # Titre sous l'image
                title_display = original_title[:25] + "..." if len(original_title) > 25 else original_title
                ax_img.text(0.5, -0.15, title_display, transform=ax_img.transAxes,
                           ha='center', va='top', fontsize=9, color='#96CEB4', style='italic')

            except Exception as e:
                ax_img.text(0.5, 0.5, f'\nErreur image #{i+1}', ha='center', va='center',
                           transform=ax_img.transAxes, fontsize=10, color='#FF6B9D')
        else:
            ax_img.text(0.5, 0.5, f'\nPas d\'image #{i+1}', ha='center', va='center',
                       transform=ax_img.transAxes, fontsize=10, color='#888888')

        ax_img.axis('off')

else:
```

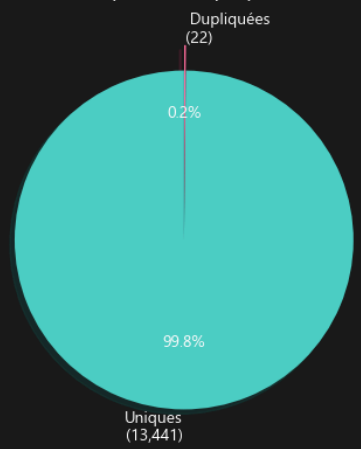
```
        # Pas de doublons complets
        ax_no_duplicates = plt.subplot(3, 1, (2, 3))
        ax_no_duplicates.set_facecolor('#2d2d2d')
        ax_no_duplicates.axis('off')
        ax_no_duplicates.text(0.5, 0.5, 'Aucun doublon complet trouvé!\nVotre dataset est très propre!',
                              ha='center', va='center', transform=ax_no_duplicates.transAxes,
                              fontsize=18, color='#4ECDC4', fontweight='bold')

    plt.tight_layout(pad=2.0, rect=[0, 0.02, 1, 0.98])
    plt.show()

    #  Nettoyage des colonnes temporaires
    recipes_with_images.drop(['title_clean_comparison', 'instructions_clean_comparison', 'combined_clean'],
                             axis=1, inplace=True)

    print("Toutes les visualisations et l'analyse des doublons sont OK!")

else:
    print("DataFrame 'recipes_with_images' non trouvé. Exécutez d'abord les cellules de chargement et de correspondance des im
```

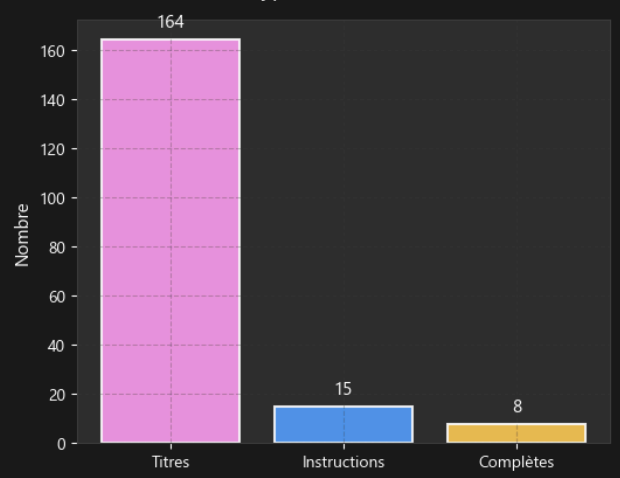
```
Analyse des top 20 ingrédients...
Top 20 ingrédients: ['kosher salt', 'sugar', 'salt', 'unsalted butter', 'olive oil', 'garlic clove', 'large egg', 'extravirgin
olive oil', 'freshly ground black pepper', 'fresh lemon juice', 'allpurpose flour', 'water', 'vegetable oil', 'vanilla extrac
t', 'powder', 'whole milk', 'freshly ground pepper', 'heavy cream', 'honey', 'fresh lime juice']
```


ANALYSE DU DATASET AVEC IMAGES
Nuage de mots - Ingrédients
Nuage de mots - Titres de Recettes
Top 20 Ingrédients
kosher salt 4736
sugar 3448
salt 2903
unsalted butter 2742
olive oil 2711
garlic clove 2540
large egg 2156
extravirgin olive oil 2068
freshly ground black pepper 1739
fresh lemon juice 1683
allpurpose flour 1503
water 1472
vegetable oil 1213
vanilla extract 958
powder 846
whole milk 827
freshly ground pepper 789
heavy cream 750
honey 719
fresh lime juice 705
Ingrédients
Fréquence
Distribution des Instructions
Moyenne: 180.9
Médiane: 153.0
Fréquence
Nombre de mots dans les instructions

```
STATISTIQUES DES INSTRUCTIONS:
Longueur moyenne: 180.9 mots
Longueur médiane: 153.0 mots
Plus courte: 6 mots
Plus longue: 2587 mots

STATISTIQUES GÉNÉRALES:
Recettes avec images: 13,463
Recettes avec ingrédients: 13,463
Recettes avec instructions: 13,463
Recettes avec nettoyage avancé: 13463

ANALYSE DES TITRES EN DOUBLE:
Titres uniques: 13,104
Titres en double: 164
Total d'occurrences dupliquées: 359
Pourcentage de doublons: 1.2%

ANALYSE DES INSTRUCTIONS EN DOUBLE:
Instructions uniques: 13,419
Instructions en double: 15
Total d'occurrences dupliquées: 44
Pourcentage de doublons: 0.1%

ANALYSE DES RECETTES COMPLÈTEMENT IDENTIQUES:
Recettes complètement uniques: 13,441
Recettes complètement identiques: 8
Total de recettes dupliquées: 22
```

ANALYSE VISUELLE DES DOUBLONS





STATISTIQUES DOUBLONS

Total recettes: 13,463
Titres dupliqués: 164
Instructions dupliquées: 15
Recettes identiques: 8

Pourcentage de doublons:
- Titres: 1.2%
- Instructions: 0.1%

EXEMPLE DE RECETTE DUPLIQUÉE (4 fois)

Titre: "manhattan"

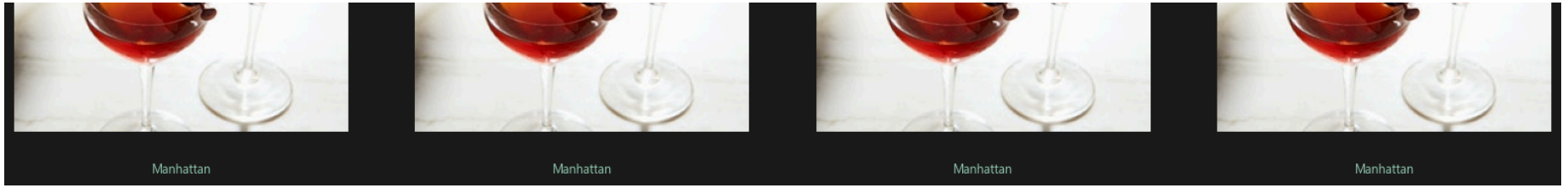
Instructions: "in mixing glass or cocktail shaker filled with ice, combine whiskey, vermouth, and bitters. stir well, about 20 seconds, then strain into cocktail gla..."

Copie #1

Copie #2

Copie #3

Copie #4




```
Toutes les visualisations et l'analyse des doublons sont OK!
```

6. Définition du modèle (EfficientNet gelé)

In [7]:
```python
#  Modèle => EfficientNet
from sklearn.metrics.pairwise import cosine_similarity
from PIL import Image

def create_image_retrieval_model(embedding_dim=1280):
    """
     Create EfficientNet model for image embedding extraction
    """
    # Load pre-trained EfficientNet
    base_model = EfficientNetB0(
        weights='imagenet',
        include_top=False,
        input_shape=(CONFIG['IMG_SIZE'], CONFIG['IMG_SIZE'], 3)
    )

    # Input shape
    inputs = Input(shape=(CONFIG['IMG_SIZE'], CONFIG['IMG_SIZE'], 3))
    x = base_model(inputs, training=False)
    x = GlobalAveragePooling2D()(x)

    # Normalisation L2 (mauvaise pratique de faire comme ça)
    def l2_norm(x):
        return tf.nn.l2_normalize(x, axis=1)

    # Same shape
    def same_shape(input_shape):
        return input_shape

    # Voir recipe_image_retrieval_tl.ipynb on a fait mieux avec des custom layers car pour charger le modèle c'est contraignan
```

```
    embeddings = Lambda(l2_norm, output_shape=same_shape, name="l2_norm")(x)

    model = Model(inputs, embeddings, name='image_retrieval_model')
    return model

# Modèle créé
image_retrieval_model = create_image_retrieval_model(CONFIG['EMBEDDING_DIM_NATIVE'])

print("Modèle chargé!")
print(f"Paramètres: {image_retrieval_model.count_params():,}")
print(f"Dimension des embeddings: {CONFIG['EMBEDDING_DIM_NATIVE']}")

# Affiché le modèle simple
image_retrieval_model.summary()
```

```
Modèle chargé!
Paramètres: 4,049,571
Dimension des embeddings: 1280
```

Model: "image_retrieval_model"

Layer (type)	Output Shape	Param #
input_layer_1 (InputLayer)	(None, 224, 224, 3)	0
efficientnetb0 (Functional)	(None, 7, 7, 1280)	4,049,571
global_average_pooling2d (GlobalAveragePooling2D)	(None, 1280)	0
l2_norm (Lambda)	(None, 1280)	0

Total params: 4,049,571 (15.45 MB)

Trainable params: 4,007,548 (15.29 MB)

Non-trainable params: 42,023 (164.16 KB)

7. Création de la classe "Recipe Image Retrieval"

In [8]:
```python
class RecipeImageRetrieval:
    """
    Système complet pour trouver les images de recettes similaires
    Input: Query image → Output: Top 3 images similaires + infos de la recette
    Utilise un EfficientNetB0 pour l'extraction d'embedding (pas d'entraînement)
    """

    def __init__(self, model, recipes_df):
        self.model = model
        self.recipes_df = recipes_df
        self.embeddings_db = None
        self.image_paths = None
        self.image_to_recipe_map = {}

    def prepare_database(self):
        """Récupération des images de recipes_with_images"""
        print("On prépare la base de données des images de recipes_with_images...")

        valid_recipes = self.recipes_df[self.recipes_df['image_path'].notna()].copy()
        self.image_paths = valid_recipes['image_path'].tolist()

        print(f"{len(self.image_paths)} images trouvées")

        # Create mapping from image path to recipe info
        for idx, row in valid_recipes.iterrows():
            self.image_to_recipe_map[row['image_path']] = {
                'title': row['Title'],
                'ingredients': row['Ingredients'],
                'instructions': row['Instructions'],
                'original_index': idx
            }

        print("Mapping image-to-recipe créé!")

    def build_embeddings_database(self, batch_size=32):
        """Extraction des embeddings de toutes les images de la base de données"""
        if self.image_paths is None:
            self.prepare_database()

        print("Extraction des embeddings...")
```

```
        embeddings = []
        total_batches = len(self.image_paths) // batch_size + (1 if len(self.image_paths) % batch_size else 0)

        for i in range(0, len(self.image_paths), batch_size):
            batch_paths = self.image_paths[i:i+batch_size]
            batch_images = []

            print(f"Batch {i//batch_size + 1}/{total_batches}...")

            for img_path in batch_paths:
                try:
                    img = tf.keras.preprocessing.image.load_img(
                        img_path, target_size=(CONFIG['IMG_SIZE'], CONFIG['IMG_SIZE'])
                    )
                    img = tf.keras.preprocessing.image.img_to_array(img)
                    img = tf.keras.applications.efficientnet.preprocess_input(img)
                    batch_images.append(img)
                except Exception as e:
                    print(f"Erreur chargement {img_path}: {e}")
                    batch_images.append(np.zeros((CONFIG['IMG_SIZE'], CONFIG['IMG_SIZE'], 3)))

            if batch_images:
                batch_images = np.array(batch_images)
                batch_embeddings = self.model.predict(batch_images, verbose=0)
                embeddings.extend(batch_embeddings)

        self.embeddings_db = np.array(embeddings)
        print(f"Base de données des embeddings créée: {len(self.embeddings_db)} embeddings")
        print(f"Shape embeddings: {self.embeddings_db.shape}")

    def visualize_model_architecture(self):
        """Visualisation de l'architecture du modèle"""
        fig = plt.figure(figsize=(16, 12))
        fig.patch.set_facecolor('#1a1a1a')

        fig.suptitle('Architecture du modèle, extraction d\'embedding (pas d\'entraînement)',
                     fontsize=18, color='white', fontweight='bold', y=0.94)

        total_params = self.model.count_params()
        trainable_params = sum(layer.count_params() for layer in self.model.layers if layer.trainable)
```

```
        frozen_params = total_params - trainable_params

        total_recipes = len(self.recipes_df)
        recipes_with_images = len(self.recipes_df[self.recipes_df['image_path'].notna()]) if 'image_path' in self.recipes_df.c
        embeddings_ready = len(self.embeddings_db) if self.embeddings_db is not None else 0

        # Distribution des paramètres (gelés et trainables), ici 100% gelés car pas d'entraînement
        ax1 = plt.subplot(2, 2, 1)
        ax1.set_facecolor('#2d2d2d')

        params_data = [frozen_params, trainable_params] if trainable_params > 0 else [total_params]
        params_labels = ['Gelés (pré-entraînés)', 'Entraînables'] if trainable_params > 0 else ['Tous gelés']
        colors_params = ['#4ECDC4', '#FF6B9D'] if trainable_params > 0 else ['#4ECDC4']

        wedges, texts, autotexts = ax1.pie(params_data, labels=params_labels, colors=colors_params,
                                           autopct='%1.1f%%', startangle=90)

        for text in texts:
            text.set_color('white')
        for autotext in autotexts:
            autotext.set_color('white')
            autotext.set_fontweight('bold')

        ax1.set_title(f'Paramètres (pas d\'entraînement)\nTotal: {total_params:,}',
                      fontsize=14, color='white', fontweight='bold', pad=15)

        # Pipeline d'extraction d'embedding
        ax2 = plt.subplot(2, 2, 2)
        ax2.set_facecolor('#2d2d2d')
        ax2.set_xlim(0, 12)
        ax2.set_ylim(0, 8)

        pipeline_steps = [
            (1.5, 5, "Image\n224x224x3", '#FF6B9D', 1.5),
            (3.8, 5, "EfficientNet\n(Gelé)", '#4ECDC4', 2.0),
            (6.5, 5, "Global Pool\n(1280-dim)", '#45B7D1', 1.8),
            (9.5, 5, "L2 Normalize\n", '#96CEB4', 1.8)
        ]

        # Dessiner les étapes
        for i, (x, y, text, color, width) in enumerate(pipeline_steps):
```

```
        ax2.add_patch(plt.Rectangle((x-width/2, y-0.9), width, 1.8,
                                    facecolor=color, alpha=0.8, edgecolor='white', linewidth=2))
        ax2.text(x, y, text, ha='center', va='center', fontsize=10,
                color='white', fontweight='bold')

        if i < len(pipeline_steps) - 1:
            next_x = pipeline_steps[i+1][0]
            next_width = pipeline_steps[i+1][4]
            ax2.arrow(x + width/2 + 0.1, y, next_x - x - width/2 - next_width/2 - 0.2, 0,
                    head_width=0.3, head_length=0.2, fc='white', ec='white', alpha=0.8)

    # Workflow explicatif
    ax2.add_patch(plt.Rectangle((0.5, 2.3), 10.5, 1.8,
                                facecolor='#1a1a1a', alpha=0.9, edgecolor='#45B7D1', linewidth=2))

    workflow_text = """
    EXTRACTION DES EMBEDDINGS: 1x ce pipeline → Extraction de 1280-dim features
    RECHERCHE: par Similarité Cosine entre l'input et la BDD des embeddings
    """

    ax2.text(5.75, 3.2, workflow_text, ha='center', va='center', fontsize=10,
            color='white', fontweight='bold')

    ax2.set_title('Pipeline d\'extraction d\'embedding', fontsize=14, color='white', fontweight='bold', pad=15)
    ax2.text(5.75, 0.8, "Pas d'entraînement - extraction d'embeddings en utilisant les poids ImageNet",
            ha='center', va='center', fontsize=9, color='#45B7D1', style='italic')
    ax2.axis('off')

    # Statistiques de la BDD des embeddings
    ax3 = plt.subplot(2, 2, 3)
    ax3.set_facecolor('#2d2d2d')

    database_stats = {
        'Total\nrecettes': total_recipes,
        'Avec\nimages': recipes_with_images,
        'Embeddings\n': embeddings_ready,
        '\nDim des embeddings': CONFIG['EMBEDDING_DIM_NATIVE']
    }

    bars = ax3.bar(list(database_stats.keys()), list(database_stats.values()),
                color=['#FF6B9D', '#4ECDC4', '#45B7D1', '#96CEB4'],
```

```
                alpha=0.9, edgecolor='white', linewidth=2)

    ax3.set_title('Statistiques de la base de données des embeddings', fontsize=14, color='white', fontweight='bold', pad=
    ax3.set_ylabel('Count', fontsize=12, color='white')
    ax3.tick_params(axis='x', labelsize=10, colors='white')
    ax3.tick_params(axis='y', colors='white')
    ax3.grid(True, alpha=0.3, color='#404040')

    for bar, value in zip(bars, database_stats.values()):
        ax3.text(bar.get_x() + bar.get_width()/2., bar.get_height() + max(database_stats.values()) * 0.01,
                f'{value:,}', ha='center', va='bottom', color='white', fontweight='bold')

    # Résumé des différents éléments du système
    ax4 = plt.subplot(2, 2, 4)
    ax4.set_facecolor('#2d2d2d')
    ax4.axis('off')

    if embeddings_ready > 0:
        status = "Prêt pour la recherche de similarité"
    elif recipes_with_images > 0:
        status = "Prêt pour l'extraction d'embedding"
    else:
        status = "Aucune image trouvée"

    combined_text = f"""
MODELE D'EXTRACTION D'EMBEDDING:

Architecture:
• Base: EfficientNetB0 (base pré-entraînée ImageNet)
• Output: {CONFIG['EMBEDDING_DIM_NATIVE']}-dim (raw)
• Traitement: GlobalAveragePooling2D + L2 normalize

Paramètres:
• Total: {total_params:,}
• Gelés: {total_params:,} (100% pré-entraînés)
• Entraînables: 0

Ce que fait ce modèle:
• Extraction d'embedding via EfficientNetB0 pré-entraîné
• GlobalAveragePooling2D → 1280 features
• L2 normalization
```

```
• Recherche par similarité Cosine

Ce que ça ne fait pas:
• Pas d'entraînement / transfer learning / fine-tuning
• Pas de custom Dense layers (tête custom)
• Pas d'optimisation spécifique aux recettes

Caractéristiques:
• Strategie: recherche de similarité Cosine
• Forces: reconnaissance correcte mais limité des images de recettes similaires
• Limites: ne comprend pas forcément toutes les subtilités propres aux recettes
• Vitesse: très rapide (pas d'entraînement), on extrait juste les embeddings

Statut: {status}
Dataset: {total_recipes:,} recettes, {recipes_with_images:,} avec images
"""

    ax4.text(0.05, 0.95, combined_text, transform=ax4.transAxes, fontsize=9,
            color='white', verticalalignment='top', fontfamily='monospace',
            bbox=dict(boxstyle="round,pad=0.5", facecolor='#2d2d2d', alpha=0.8))

    plt.tight_layout(pad=2.0, rect=[0, 0.02, 1, 0.92])
    plt.show()

def visualize_embedding_space(self, sample_size=500):
    """ Visualisation simplifiée des features EfficientNet """
    if self.embeddings_db is None:
        print("Il faut d'abord créer la base de données des embeddings!")
        return

    if len(self.embeddings_db) > sample_size:
        indices = np.random.choice(len(self.embeddings_db), sample_size, replace=False)
        sample_embeddings = self.embeddings_db[indices]
    else:
        sample_embeddings = self.embeddings_db
        indices = np.arange(len(sample_embeddings))

    # Création des plots
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 6))
    fig.patch.set_facecolor('#1a1a1a')
    fig.suptitle('Analyse des features EfficientNet - Modèle Raw',
```

```
                fontsize=18, color='white', fontweight='bold', y=0.95)

    # Plot 1: Distribution des features
    ax1.hist(sample_embeddings.flatten(), bins=50, alpha=0.7, color='skyblue', edgecolor='black')
    ax1.set_facecolor('#2d2d2d')
    ax1.set_title('Distribution des features (1280D)', fontsize=14, color='white', fontweight='bold')
    ax1.set_xlabel('Valeurs des features', color='white')
    ax1.set_ylabel('Fréquence', color='white')
    ax1.tick_params(colors='white')
    ax1.grid(True, alpha=0.3, color='#404040')

    # Plot 2: Clustering visuel avec PCA
    from sklearn.decomposition import PCA

    pca = PCA(n_components=2)
    embeddings_2d = pca.fit_transform(sample_embeddings)

    # Analyser les propriétés visuelles basiques
    image_properties = []
    for i in indices:
        try:
            img = Image.open(self.image_paths[i])
            img_array = np.array(img)
            brightness = np.mean(img_array) / 255.0
            image_properties.append(brightness)
        except:
            image_properties.append(0.5)

    # Colorier par brightness
    scatter = ax2.scatter(embeddings_2d[:, 0], embeddings_2d[:, 1],
                          c=image_properties, cmap='viridis', s=50, alpha=0.7)

    ax2.set_facecolor('#2d2d2d')
    ax2.set_title('Clusters visuels (PCA)', fontsize=14, color='white', fontweight='bold')
    ax2.set_xlabel(f'PC1 ({pca.explained_variance_ratio_[0]:.1%})', color='white')
    ax2.set_ylabel(f'PC2 ({pca.explained_variance_ratio_[1]:.1%})', color='white')
    ax2.tick_params(colors='white')
    ax2.grid(True, alpha=0.3, color='#404040')

    plt.colorbar(scatter, ax=ax2, label='Luminosité')
```

```
        plt.tight_layout()
        plt.show()

        # Statistiques simples
        print(f"\nStatistiques:")
        print(f"Échantillons analysés: {len(sample_embeddings)}")
        print(f"Dimensions: {sample_embeddings.shape[1]} features EfficientNet")
        print(f"Variance expliquée: {sum(pca.explained_variance_ratio_):.1%}")
        print(f"Luminosité moyenne: {np.mean(image_properties):.2f}")

        print("\nNote: Features génériques ImageNet sans apprentissage spécifique aux recettes")

    def visualize_similarity_matrix(self, sample_size=20):
        """Visualisation de la matrice de similarité entre les recettes"""
        if self.embeddings_db is None:
            print("Il faut d'abord créer la base de données des embeddings!")
            return

        # Test sur des recettes aléatoires
        indices = np.random.choice(len(self.embeddings_db), min(sample_size, len(self.embeddings_db)), replace=False)
        sample_embeddings = self.embeddings_db[indices]
        sample_titles = [self.image_to_recipe_map[self.image_paths[i]]['title'][:20] + "..."
                         for i in indices]

        # Calculate similarity matrix
        similarity_matrix = cosine_similarity(sample_embeddings)

        # Heatmap
        fig, ax = plt.subplots(figsize=(15, 12))
        fig.patch.set_facecolor('#1a1a1a')

        im = ax.imshow(similarity_matrix, cmap='viridis', aspect='auto')

        cbar = plt.colorbar(im, ax=ax)
        cbar.ax.tick_params(colors='white')
        cbar.set_label('Similarité Cosine', color='white', fontweight='bold')

        ax.set_xticks(range(len(sample_titles)))
        ax.set_yticks(range(len(sample_titles)))
        ax.set_xticklabels(sample_titles, rotation=45, ha='right', color='white', fontsize=9)
        ax.set_yticklabels(sample_titles, color='white', fontsize=9)
```

```
        # Add text annotations
        for i in range(len(sample_titles)):
            for j in range(len(sample_titles)):
                text = ax.text(j, i, f'{similarity_matrix[i, j]:.2f}',
                               ha="center", va="center", color="white", fontsize=8, fontweight='bold')

        ax.set_title('MATRICE DE SIMILARITE',
                     fontsize=16, color='white', fontweight='bold', pad=20)
        ax.set_facecolor('#2d2d2d')

        plt.tight_layout()
        plt.show()

        print(f"Matrice de similarité pour {len(sample_titles)} recettes")

    # Core functionality (unchanged)
    def search_similar_recipes(self, query_image_path, top_k=3):
        """Rechercher le top-k des recettes similaires à une image de recette"""
        if self.embeddings_db is None:
            print("Il faut d'abord créer la base de données des embeddings!")
            return None

        print(f"Recherche de {top_k} recettes similaires...")

        try:
            img = tf.keras.preprocessing.image.load_img(
                query_image_path, target_size=(CONFIG['IMG_SIZE'], CONFIG['IMG_SIZE'])
            )
            img = tf.keras.preprocessing.image.img_to_array(img)
            img = tf.keras.applications.efficientnet.preprocess_input(img)
            img = np.expand_dims(img, axis=0)

            query_embedding = self.model.predict(img, verbose=0)
        except Exception as e:
            print(f"Erreur chargement de l'image de recette: {e}")
            return None

        # Calcul des similarités
        similarities = cosine_similarity(query_embedding, self.embeddings_db)[0]
```

```
        # Get top k
        top_indices = np.argsort(similarities)[::-1][:top_k]

        results = []
        for rank, idx in enumerate(top_indices):
            img_path = self.image_paths[idx]
            similarity_score = similarities[idx]

            if img_path in self.image_to_recipe_map:
                recipe_info = self.image_to_recipe_map[img_path].copy()
                recipe_info['rank'] = rank + 1
                recipe_info['similarity'] = similarity_score
                recipe_info['similar_image_path'] = img_path
                results.append(recipe_info)

        print(f"{len(results)} recipes similaires trouvées")
        return results

    def display_results(self, query_image_path, results):
        """Afficher les résultats de la recherche"""
        if not results:
            print("Aucun résultat")
            return

        fig, axes = plt.subplots(1, len(results) + 1, figsize=(20, 6))
        fig.patch.set_facecolor('#1a1a1a')
        fig.suptitle('IMAGE → TOP 3 RECETTES SIMILAIRES',
                     fontsize=18, color='white', fontweight='bold', y=0.95)

        colors = ['#FF6B9D', '#4ECDC4', '#45B7D1', '#96CEB4']

        # Image en input
        try:
            query_img = Image.open(query_image_path)
            axes[0].imshow(query_img)
            axes[0].set_title("INPUT", fontsize=14, color=colors[0], fontweight='bold', pad=15)
            axes[0].axis('off')

            # Add border
            for spine in axes[0].spines.values():
                spine.set_visible(True)
```

```
            spine.set_color(colors[0])
            spine.set_linewidth(3)
    except:
        axes[0].text(0.5, 0.5, "Pas d\'input", ha='center', va='center',
                     transform=axes[0].transAxes, color='white', fontsize=12)
        axes[0].set_title("INPUT", fontsize=14, color=colors[0], fontweight='bold')
        axes[0].axis('off')

    # Similar images
    for i, result in enumerate(results):
        try:
            result_img = Image.open(result['similar_image_path'])
            axes[i+1].imshow(result_img)

            title = result['title'][:25] + "..." if len(result['title']) > 25 else result['title']
            similarity = result['similarity']

            axes[i+1].set_title(f"#{result['rank']}: {title}\n Similarity: {similarity:.3f}",
                                fontsize=12, color=colors[i+1], fontweight='bold', pad=15)
            axes[i+1].axis('off')

            # Add colored border
            for spine in axes[i+1].spines.values():
                spine.set_visible(True)
                spine.set_color(colors[i+1])
                spine.set_linewidth(3)

        except:
            axes[i+1].text(0.5, 0.5, f"Image #{result['rank']}\nabsente",
                           ha='center', va='center', transform=axes[i+1].transAxes,
                           color='white', fontsize=11)
            axes[i+1].set_title(f"#{result['rank']}: ERREUR", fontsize=12, color='red', fontweight='bold')
            axes[i+1].axis('off')

    plt.tight_layout()
    plt.show()

    print("\n" + "="*80)
    print("RECETTES SIMILAIRES")
    print("="*80)
```

```
        for result in results:
            print(f"\nRANG #{result['rank']}: {result['similarity']:.4f}")
            print(f"{result['title']}")
            print("-" * 60)

            # Ingredients
            ingredients = str(result['ingredients'])
            if len(ingredients) > 300:
                ingredients = ingredients[:300] + "..."
            print(f"Ingredients:\n{ingredients}")
            print("-" * 60)

            # Instructions
            instructions = str(result['instructions'])
            if len(instructions) > 400:
                instructions = instructions[:400] + "..."
            print(f"Instructions:\n{instructions}")
            print("=" * 80)

    def test_with_random_image(self, top_k=3):
        """Test avec une image aléatoire de la base de données"""
        if not self.image_paths:
            print("Aucune image disponible")
            return

        random_idx = np.random.randint(0, len(self.image_paths))
        test_image_path = self.image_paths[random_idx]

        print(f"Test avec une image aléatoire: {test_image_path}")

        results = self.search_similar_recipes(test_image_path, top_k)

        if results:
            self.display_results(test_image_path, results)
        else:
            print("Aucun résultat")

print("Système créé!")
print("Cette classe permet d'extraire les embeddings des images de recettes et de les rechercher par similarité")
print("\nMéthodes disponibles:")
print("- retrieval_system.visualize_model_architecture()")
```

```
print("- retrieval_system.visualize_embedding_space()")
print("- retrieval_system.visualize_similarity_matrix()")
```

```
Système créé!
Cette classe permet d'extraire les embeddings des images de recettes et de les rechercher par similarité

Méthodes disponibles:
- retrieval_system.visualize_model_architecture()
- retrieval_system.visualize_embedding_space()
- retrieval_system.visualize_similarity_matrix()
```

8. Extraction des embeddings depuis le dataset

In [9]:
```
#  Extraction des features
print("Initialisation du système...")

# Check if recipes_with_images exists
if 'recipes_with_images' in globals() and len(recipes_with_images) > 0:
    print(f"{len(recipes_with_images)} recettes avec images")

    retrieval_system = RecipeImageRetrieval(image_retrieval_model, recipes_with_images)
    retrieval_system.prepare_database()

    print("Système initialisé avec succès!")
    print(f"La base de données contient {len(retrieval_system.image_paths)} images")

    sample_titles = [retrieval_system.image_to_recipe_map[path]['title']
                     for path in list(retrieval_system.image_paths)[:5]]
    print(f"Exemples de recettes: {sample_titles}")

else:
    print("'recipes_with_images' n'est pas défini!")
    print("Veuillez d'abord exécuter les cellules précédentes pour charger et traiter les données.")

#  Extraction des embeddings
print("Extraction des embeddings...")
retrieval_system.build_embeddings_database(batch_size=16)

print("\nProchaines étapes:")
```

```
print("1. Test sur une image aléatoire du dataset: retrieval_system.test_with_random_image()")
print("2. Test sur une image de recette: retrieval_system.search_similar_recipes('path/to/image.jpg')")
```

```
Initialisation du système...
13463 recettes avec images
On prépare la base de données des images de recipes_with_images...
13463 images trouvées
Mapping image-to-recipe créé!
Système initialisé avec succès!
La base de données contient 13463 images
Exemples de recettes: ['Miso-Butter Roast Chicken With Acorn Squash Panzanella', 'Crispy Salt and Pepper Potatoes', 'Thanksgivi
ng Mac and Cheese', 'Italian Sausage and Bread Stuffing', "Newton's Law"]
Extraction des embeddings...
Extraction des embeddings...
Batch 1/842...
Batch 2/842...
Batch 3/842...
Batch 4/842...
Batch 5/842...
Batch 6/842...
Batch 7/842...
Batch 8/842...
Batch 9/842...
Batch 10/842...
Batch 11/842...
Batch 12/842...
Batch 13/842...
Batch 14/842...
Batch 15/842...
Batch 16/842...
Batch 17/842...
Batch 18/842...
Batch 19/842...
Batch 20/842...
Batch 21/842...
Batch 22/842...
Batch 23/842...
Batch 24/842...
Batch 25/842...
Batch 26/842...
Batch 27/842...
Batch 28/842...
Batch 29/842...
Batch 30/842...
```

```
Batch 31/842...
Batch 32/842...
Batch 33/842...
Batch 34/842...
Batch 35/842...
Batch 36/842...
Batch 37/842...
Batch 38/842...
Batch 39/842...
Batch 40/842...
Batch 41/842...
Batch 42/842...
Batch 43/842...
Batch 44/842...
Batch 45/842...
Batch 46/842...
Batch 47/842...
Batch 48/842...
Batch 49/842...
Batch 50/842...
Batch 51/842...
Batch 52/842...
Batch 53/842...
Batch 54/842...
Batch 55/842...
Batch 56/842...
Batch 57/842...
Batch 58/842...
Batch 59/842...
Batch 60/842...
Batch 61/842...
Batch 62/842...
Batch 63/842...
Batch 64/842...
Batch 65/842...
Batch 66/842...
Batch 67/842...
Batch 68/842...
Batch 69/842...
Batch 70/842...
Batch 71/842...
```

```
Batch 72/842...
Batch 73/842...
Batch 74/842...
Batch 75/842...
Batch 76/842...
Batch 77/842...
Batch 78/842...
Batch 79/842...
Batch 80/842...
Batch 81/842...
Batch 82/842...
Batch 83/842...
Batch 84/842...
Batch 85/842...
Batch 86/842...
Batch 87/842...
Batch 88/842...
Batch 89/842...
Batch 90/842...
Batch 91/842...
Batch 92/842...
Batch 93/842...
Batch 94/842...
Batch 95/842...
Batch 96/842...
Batch 97/842...
Batch 98/842...
Batch 99/842...
Batch 100/842...
Batch 101/842...
Batch 102/842...
Batch 103/842...
Batch 104/842...
Batch 105/842...
Batch 106/842...
Batch 107/842...
Batch 108/842...
Batch 109/842...
Batch 110/842...
Batch 111/842...
Batch 112/842...
```

```
Batch 113/842...
Batch 114/842...
Batch 115/842...
Batch 116/842...
Batch 117/842...
Batch 118/842...
Batch 119/842...
Batch 120/842...
Batch 121/842...
Batch 122/842...
Batch 123/842...
Batch 124/842...
Batch 125/842...
Batch 126/842...
Batch 127/842...
Batch 128/842...
Batch 129/842...
Batch 130/842...
Batch 131/842...
Batch 132/842...
Batch 133/842...
Batch 134/842...
Batch 135/842...
Batch 136/842...
Batch 137/842...
Batch 138/842...
Batch 139/842...
Batch 140/842...
Batch 141/842...
Batch 142/842...
Batch 143/842...
Batch 144/842...
Batch 145/842...
Batch 146/842...
Batch 147/842...
Batch 148/842...
Batch 149/842...
Batch 150/842...
Batch 151/842...
Batch 152/842...
Batch 153/842...
```

```
Batch 154/842...
Batch 155/842...
Batch 156/842...
Batch 157/842...
Batch 158/842...
Batch 159/842...
Batch 160/842...
Batch 161/842...
Batch 162/842...
Batch 163/842...
Batch 164/842...
Batch 165/842...
Batch 166/842...
Batch 167/842...
Batch 168/842...
Batch 169/842...
Batch 170/842...
Batch 171/842...
Batch 172/842...
Batch 173/842...
Batch 174/842...
Batch 175/842...
Batch 176/842...
Batch 177/842...
Batch 178/842...
Batch 179/842...
Batch 180/842...
Batch 181/842...
Batch 182/842...
Batch 183/842...
Batch 184/842...
Batch 185/842...
Batch 186/842...
Batch 187/842...
Batch 188/842...
Batch 189/842...
Batch 190/842...
Batch 191/842...
Batch 192/842...
Batch 193/842...
Batch 194/842...
```

```
Batch 195/842...
Batch 196/842...
Batch 197/842...
Batch 198/842...
Batch 199/842...
Batch 200/842...
Batch 201/842...
Batch 202/842...
Batch 203/842...
Batch 204/842...
Batch 205/842...
Batch 206/842...
Batch 207/842...
Batch 208/842...
Batch 209/842...
Batch 210/842...
Batch 211/842...
Batch 212/842...
Batch 213/842...
Batch 214/842...
Batch 215/842...
Batch 216/842...
Batch 217/842...
Batch 218/842...
Batch 219/842...
Batch 220/842...
Batch 221/842...
Batch 222/842...
Batch 223/842...
Batch 224/842...
Batch 225/842...
Batch 226/842...
Batch 227/842...
Batch 228/842...
Batch 229/842...
Batch 230/842...
Batch 231/842...
Batch 232/842...
Batch 233/842...
Batch 234/842...
Batch 235/842...
```

```
Batch 236/842...
Batch 237/842...
Batch 238/842...
Batch 239/842...
Batch 240/842...
Batch 241/842...
Batch 242/842...
Batch 243/842...
Batch 244/842...
Batch 245/842...
Batch 246/842...
Batch 247/842...
Batch 248/842...
Batch 249/842...
Batch 250/842...
Batch 251/842...
Batch 252/842...
Batch 253/842...
Batch 254/842...
Batch 255/842...
Batch 256/842...
Batch 257/842...
Batch 258/842...
Batch 259/842...
Batch 260/842...
Batch 261/842...
Batch 262/842...
Batch 263/842...
Batch 264/842...
Batch 265/842...
Batch 266/842...
Batch 267/842...
Batch 268/842...
Batch 269/842...
Batch 270/842...
Batch 271/842...
Batch 272/842...
Batch 273/842...
Batch 274/842...
Batch 275/842...
Batch 276/842...
```

```
Batch 277/842...
Batch 278/842...
Batch 279/842...
Batch 280/842...
Batch 281/842...
Batch 282/842...
Batch 283/842...
Batch 284/842...
Batch 285/842...
Batch 286/842...
Batch 287/842...
Batch 288/842...
Batch 289/842...
Batch 290/842...
Batch 291/842...
Batch 292/842...
Batch 293/842...
Batch 294/842...
Batch 295/842...
Batch 296/842...
Batch 297/842...
Batch 298/842...
Batch 299/842...
Batch 300/842...
Batch 301/842...
Batch 302/842...
Batch 303/842...
Batch 304/842...
Batch 305/842...
Batch 306/842...
Batch 307/842...
Batch 308/842...
Batch 309/842...
Batch 310/842...
Batch 311/842...
Batch 312/842...
Batch 313/842...
Batch 314/842...
Batch 315/842...
Batch 316/842...
Batch 317/842...
```

```
Batch 318/842...
Batch 319/842...
Batch 320/842...
Batch 321/842...
Batch 322/842...
Batch 323/842...
Batch 324/842...
Batch 325/842...
Batch 326/842...
Batch 327/842...
Batch 328/842...
Batch 329/842...
Batch 330/842...
Batch 331/842...
Batch 332/842...
Batch 333/842...
Batch 334/842...
Batch 335/842...
Batch 336/842...
Batch 337/842...
Batch 338/842...
Batch 339/842...
Batch 340/842...
Batch 341/842...
Batch 342/842...
Batch 343/842...
Batch 344/842...
Batch 345/842...
Batch 346/842...
Batch 347/842...
Batch 348/842...
Batch 349/842...
Batch 350/842...
Batch 351/842...
Batch 352/842...
Batch 353/842...
Batch 354/842...
Batch 355/842...
Batch 356/842...
Batch 357/842...
Batch 358/842...
```

```
Batch 359/842...
Batch 360/842...
Batch 361/842...
Batch 362/842...
Batch 363/842...
Batch 364/842...
Batch 365/842...
Batch 366/842...
Batch 367/842...
Batch 368/842...
Batch 369/842...
Batch 370/842...
Batch 371/842...
Batch 372/842...
Batch 373/842...
Batch 374/842...
Batch 375/842...
Batch 376/842...
Batch 377/842...
Batch 378/842...
Batch 379/842...
Batch 380/842...
Batch 381/842...
Batch 382/842...
Batch 383/842...
Batch 384/842...
Batch 385/842...
Batch 386/842...
Batch 387/842...
Batch 388/842...
Batch 389/842...
Batch 390/842...
Batch 391/842...
Batch 392/842...
Batch 393/842...
Batch 394/842...
Batch 395/842...
Batch 396/842...
Batch 397/842...
Batch 398/842...
Batch 399/842...
```

```
Batch 400/842...
Batch 401/842...
Batch 402/842...
Batch 403/842...
Batch 404/842...
Batch 405/842...
Batch 406/842...
Batch 407/842...
Batch 408/842...
Batch 409/842...
Batch 410/842...
Batch 411/842...
Batch 412/842...
Batch 413/842...
Batch 414/842...
Batch 415/842...
Batch 416/842...
Batch 417/842...
Batch 418/842...
Batch 419/842...
Batch 420/842...
Batch 421/842...
Batch 422/842...
Batch 423/842...
Batch 424/842...
Batch 425/842...
Batch 426/842...
Batch 427/842...
Batch 428/842...
Batch 429/842...
Batch 430/842...
Batch 431/842...
Batch 432/842...
Batch 433/842...
Batch 434/842...
Batch 435/842...
Batch 436/842...
Batch 437/842...
Batch 438/842...
Batch 439/842...
Batch 440/842...
```

```
Batch 441/842...
Batch 442/842...
Batch 443/842...
Batch 444/842...
Batch 445/842...
Batch 446/842...
Batch 447/842...
Batch 448/842...
Batch 449/842...
Batch 450/842...
Batch 451/842...
Batch 452/842...
Batch 453/842...
Batch 454/842...
Batch 455/842...
Batch 456/842...
Batch 457/842...
Batch 458/842...
Batch 459/842...
Batch 460/842...
Batch 461/842...
Batch 462/842...
Batch 463/842...
Batch 464/842...
Batch 465/842...
Batch 466/842...
Batch 467/842...
Batch 468/842...
Batch 469/842...
Batch 470/842...
Batch 471/842...
Batch 472/842...
Batch 473/842...
Batch 474/842...
Batch 475/842...
Batch 476/842...
Batch 477/842...
Batch 478/842...
Batch 479/842...
Batch 480/842...
Batch 481/842...
```

```
Batch 482/842...
Batch 483/842...
Batch 484/842...
Batch 485/842...
Batch 486/842...
Batch 487/842...
Batch 488/842...
Batch 489/842...
Batch 490/842...
Batch 491/842...
Batch 492/842...
Batch 493/842...
Batch 494/842...
Batch 495/842...
Batch 496/842...
Batch 497/842...
Batch 498/842...
Batch 499/842...
Batch 500/842...
Batch 501/842...
Batch 502/842...
Batch 503/842...
Batch 504/842...
Batch 505/842...
Batch 506/842...
Batch 507/842...
Batch 508/842...
Batch 509/842...
Batch 510/842...
Batch 511/842...
Batch 512/842...
Batch 513/842...
Batch 514/842...
Batch 515/842...
Batch 516/842...
Batch 517/842...
Batch 518/842...
Batch 519/842...
Batch 520/842...
Batch 521/842...
Batch 522/842...
```

```
Batch 523/842...
Batch 524/842...
Batch 525/842...
Batch 526/842...
Batch 527/842...
Batch 528/842...
Batch 529/842...
Batch 530/842...
Batch 531/842...
Batch 532/842...
Batch 533/842...
Batch 534/842...
Batch 535/842...
Batch 536/842...
Batch 537/842...
Batch 538/842...
Batch 539/842...
Batch 540/842...
Batch 541/842...
Batch 542/842...
Batch 543/842...
Batch 544/842...
Batch 545/842...
Batch 546/842...
Batch 547/842...
Batch 548/842...
Batch 549/842...
Batch 550/842...
Batch 551/842...
Batch 552/842...
Batch 553/842...
Batch 554/842...
Batch 555/842...
Batch 556/842...
Batch 557/842...
Batch 558/842...
Batch 559/842...
Batch 560/842...
Batch 561/842...
Batch 562/842...
Batch 563/842...
```

```
Batch 564/842...
Batch 565/842...
Batch 566/842...
Batch 567/842...
Batch 568/842...
Batch 569/842...
Batch 570/842...
Batch 571/842...
Batch 572/842...
Batch 573/842...
Batch 574/842...
Batch 575/842...
Batch 576/842...
Batch 577/842...
Batch 578/842...
Batch 579/842...
Batch 580/842...
Batch 581/842...
Batch 582/842...
Batch 583/842...
Batch 584/842...
Batch 585/842...
Batch 586/842...
Batch 587/842...
Batch 588/842...
Batch 589/842...
Batch 590/842...
Batch 591/842...
Batch 592/842...
Batch 593/842...
Batch 594/842...
Batch 595/842...
Batch 596/842...
Batch 597/842...
Batch 598/842...
Batch 599/842...
Batch 600/842...
Batch 601/842...
Batch 602/842...
Batch 603/842...
Batch 604/842...
```

```
Batch 605/842...
Batch 606/842...
Batch 607/842...
Batch 608/842...
Batch 609/842...
Batch 610/842...
Batch 611/842...
Batch 612/842...
Batch 613/842...
Batch 614/842...
Batch 615/842...
Batch 616/842...
Batch 617/842...
Batch 618/842...
Batch 619/842...
Batch 620/842...
Batch 621/842...
Batch 622/842...
Batch 623/842...
Batch 624/842...
Batch 625/842...
Batch 626/842...
Batch 627/842...
Batch 628/842...
Batch 629/842...
Batch 630/842...
Batch 631/842...
Batch 632/842...
Batch 633/842...
Batch 634/842...
Batch 635/842...
Batch 636/842...
Batch 637/842...
Batch 638/842...
Batch 639/842...
Batch 640/842...
Batch 641/842...
Batch 642/842...
Batch 643/842...
Batch 644/842...
Batch 645/842...
```

```
Batch 646/842...
Batch 647/842...
Batch 648/842...
Batch 649/842...
Batch 650/842...
Batch 651/842...
Batch 652/842...
Batch 653/842...
Batch 654/842...
Batch 655/842...
Batch 656/842...
Batch 657/842...
Batch 658/842...
Batch 659/842...
Batch 660/842...
Batch 661/842...
Batch 662/842...
Batch 663/842...
Batch 664/842...
Batch 665/842...
Batch 666/842...
Batch 667/842...
Batch 668/842...
Batch 669/842...
Batch 670/842...
Batch 671/842...
Batch 672/842...
Batch 673/842...
Batch 674/842...
Batch 675/842...
Batch 676/842...
Batch 677/842...
Batch 678/842...
Batch 679/842...
Batch 680/842...
Batch 681/842...
Batch 682/842...
Batch 683/842...
Batch 684/842...
Batch 685/842...
Batch 686/842...
```

```
Batch 687/842...
Batch 688/842...
Batch 689/842...
Batch 690/842...
Batch 691/842...
Batch 692/842...
Batch 693/842...
Batch 694/842...
Batch 695/842...
Batch 696/842...
Batch 697/842...
Batch 698/842...
Batch 699/842...
Batch 700/842...
Batch 701/842...
Batch 702/842...
Batch 703/842...
Batch 704/842...
Batch 705/842...
Batch 706/842...
Batch 707/842...
Batch 708/842...
Batch 709/842...
Batch 710/842...
Batch 711/842...
Batch 712/842...
Batch 713/842...
Batch 714/842...
Batch 715/842...
Batch 716/842...
Batch 717/842...
Batch 718/842...
Batch 719/842...
Batch 720/842...
Batch 721/842...
Batch 722/842...
Batch 723/842...
Batch 724/842...
Batch 725/842...
Batch 726/842...
Batch 727/842...
```

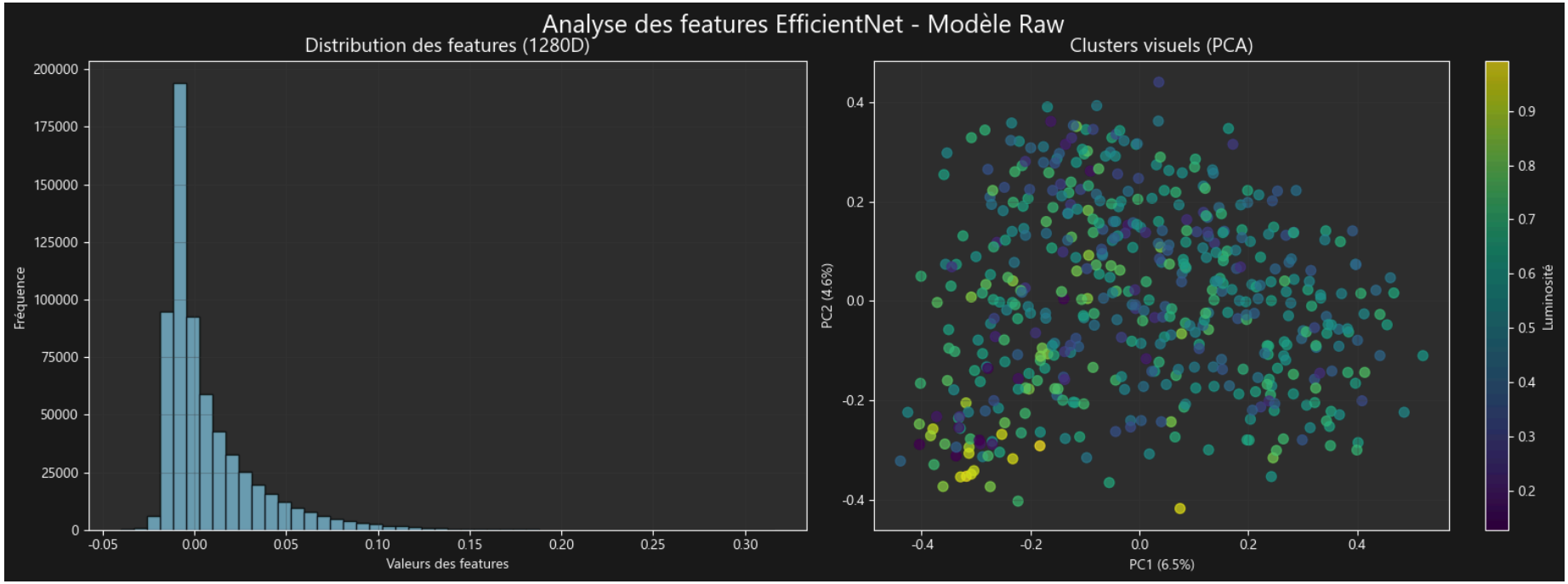
```
Batch 728/842...
Batch 729/842...
Batch 730/842...
Batch 731/842...
Batch 732/842...
Batch 733/842...
Batch 734/842...
Batch 735/842...
Batch 736/842...
Batch 737/842...
Batch 738/842...
Batch 739/842...
Batch 740/842...
Batch 741/842...
Batch 742/842...
Batch 743/842...
Batch 744/842...
Batch 745/842...
Batch 746/842...
Batch 747/842...
Batch 748/842...
Batch 749/842...
Batch 750/842...
Batch 751/842...
Batch 752/842...
Batch 753/842...
Batch 754/842...
Batch 755/842...
Batch 756/842...
Batch 757/842...
Batch 758/842...
Batch 759/842...
Batch 760/842...
Batch 761/842...
Batch 762/842...
Batch 763/842...
Batch 764/842...
Batch 765/842...
Batch 766/842...
Batch 767/842...
Batch 768/842...
```

```
Batch 769/842...
Batch 770/842...
Batch 771/842...
Batch 772/842...
Batch 773/842...
Batch 774/842...
Batch 775/842...
Batch 776/842...
Batch 777/842...
Batch 778/842...
Batch 779/842...
Batch 780/842...
Batch 781/842...
Batch 782/842...
Batch 783/842...
Batch 784/842...
Batch 785/842...
Batch 786/842...
Batch 787/842...
Batch 788/842...
Batch 789/842...
Batch 790/842...
Batch 791/842...
Batch 792/842...
Batch 793/842...
Batch 794/842...
Batch 795/842...
Batch 796/842...
Batch 797/842...
Batch 798/842...
Batch 799/842...
Batch 800/842...
Batch 801/842...
Batch 802/842...
Batch 803/842...
Batch 804/842...
Batch 805/842...
Batch 806/842...
Batch 807/842...
Batch 808/842...
Batch 809/842...
```

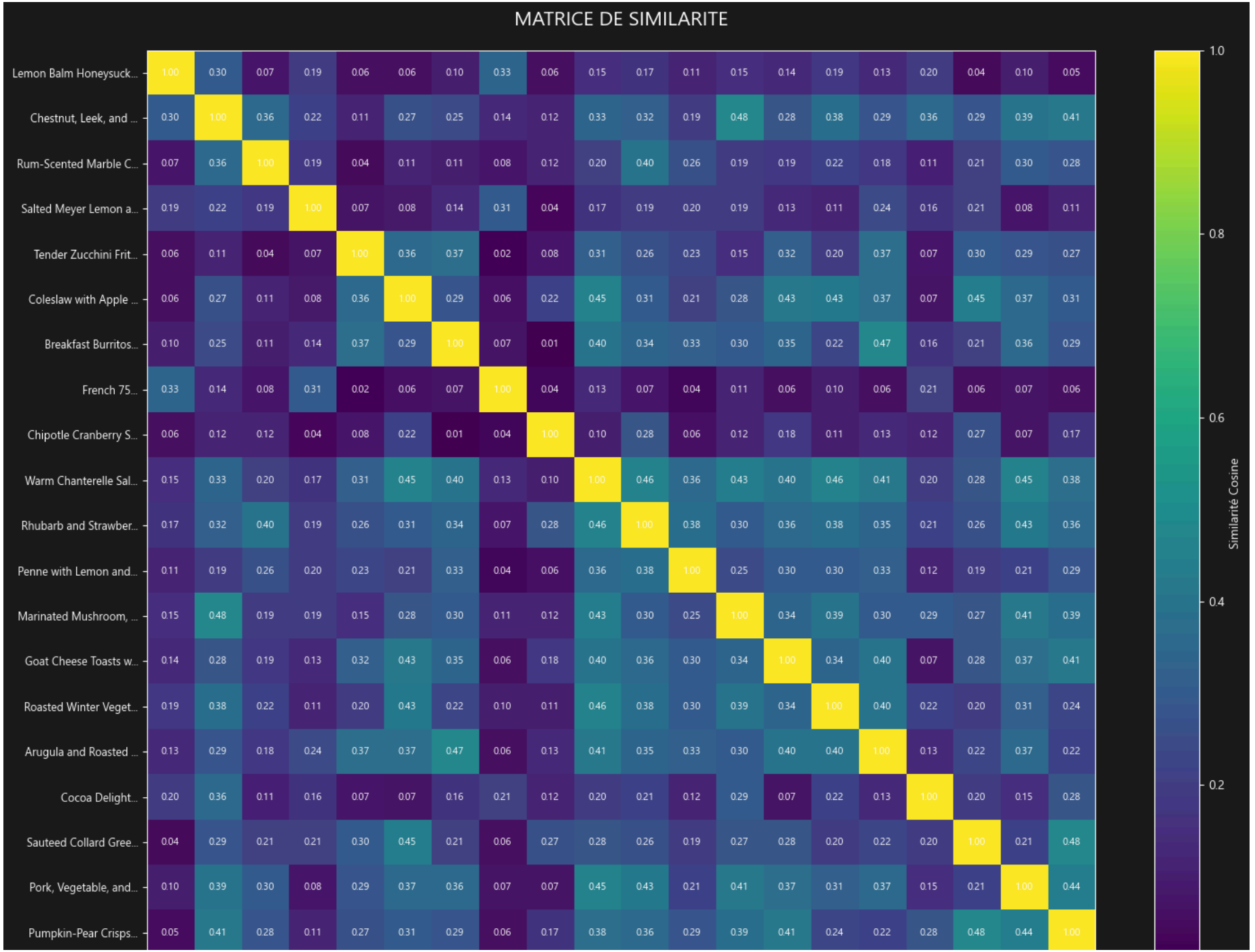
```
Batch 810/842...
Batch 811/842...
Batch 812/842...
Batch 813/842...
Batch 814/842...
Batch 815/842...
Batch 816/842...
Batch 817/842...
Batch 818/842...
Batch 819/842...
Batch 820/842...
Batch 821/842...
Batch 822/842...
Batch 823/842...
Batch 824/842...
Batch 825/842...
Batch 826/842...
Batch 827/842...
Batch 828/842...
Batch 829/842...
Batch 830/842...
Batch 831/842...
Batch 832/842...
Batch 833/842...
Batch 834/842...
Batch 835/842...
Batch 836/842...
Batch 837/842...
Batch 838/842...
Batch 839/842...
Batch 840/842...
Batch 841/842...
Batch 842/842...
Base de données des embeddings créée: 13463 embeddings
Shape embeddings: (13463, 1280)

Prochaines étapes:
1. Test sur une image aléatoire du dataset: retrieval_system.test_with_random_image()
2. Test sur une image de recette: retrieval_system.search_similar_recipes('path/to/image.jpg')
```

9. Visualisations des caractéristiques du modèle et des embeddings extraits

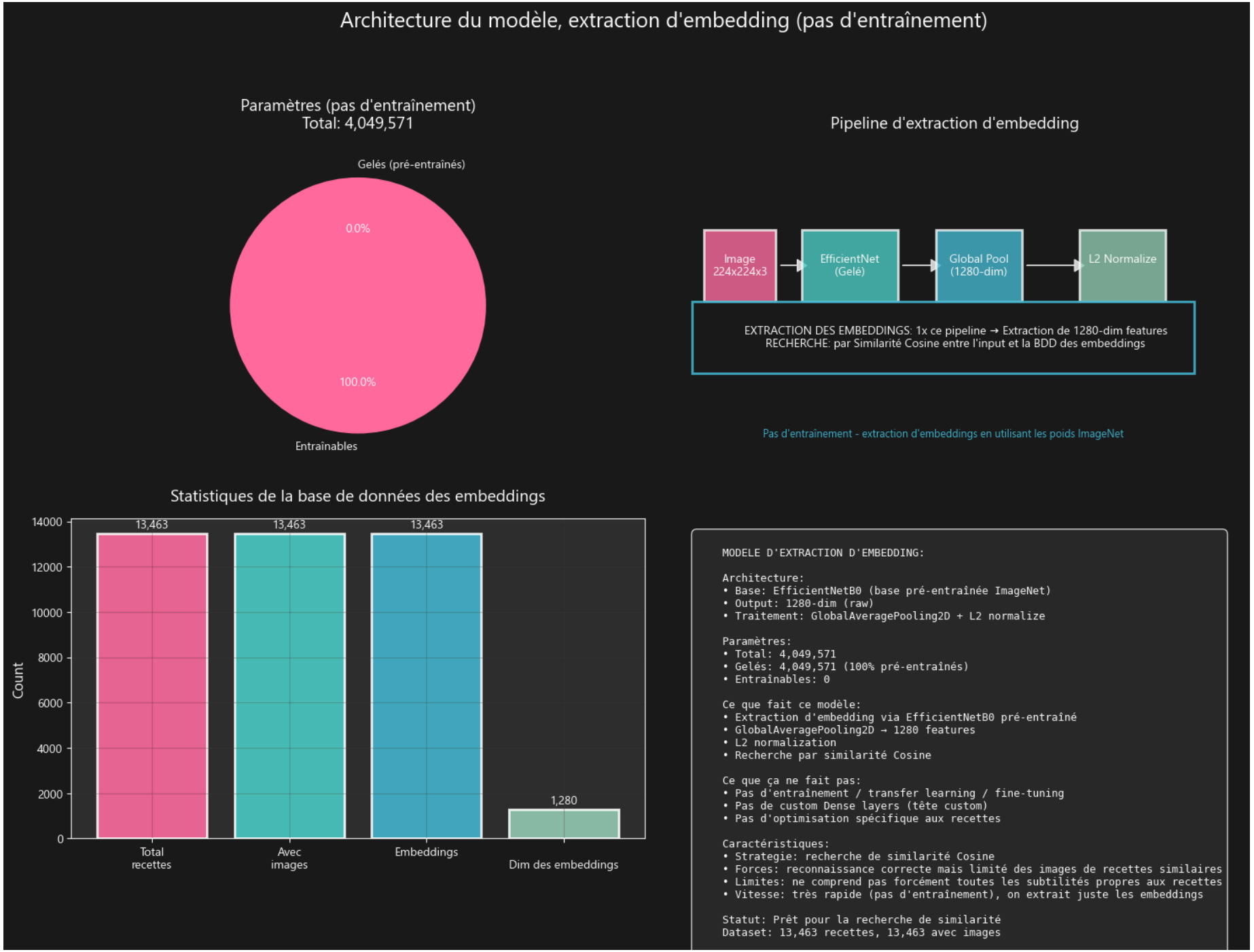
In [10]:
```
retrieval_system.visualize_embedding_space()
retrieval_system.visualize_similarity_matrix()
retrieval_system.visualize_model_architecture()
```




```
Statistiques:
Échantillons analysés: 500
Dimensions: 1280 features EfficientNet
Variance expliquée: 11.1%
Luminosité moyenne: 0.56

Note: Features génériques ImageNet sans apprentissage spécifique aux recettes
```


MATRICE DE SIMILARITE
Lemon Balm Honeysuck... 1.00 0.30 0.07 0.19 0.06 0.06 0.10 0.33 0.06 0.15 0.17 0.11 0.15 0.14 0.19 0.13 0.20 0.04 0.10 0.05
Chestnut, Leek, and ... 0.30 1.00 0.36 0.22 0.11 0.27 0.25 0.14 0.12 0.33 0.32 0.19 0.48 0.28 0.38 0.29 0.36 0.29 0.39 0.41
Rum-Scented Marble C... 0.07 0.36 1.00 0.19 0.04 0.11 0.11 0.08 0.12 0.20 0.40 0.26 0.19 0.19 0.22 0.18 0.11 0.21 0.30 0.28
Salted Meyer Lemon a... 0.19 0.22 0.19 1.00 0.07 0.08 0.14 0.31 0.04 0.17 0.19 0.20 0.19 0.13 0.11 0.24 0.16 0.21 0.08 0.11
Tender Zucchini Frit... 0.06 0.11 0.04 0.07 1.00 0.36 0.37 0.02 0.08 0.31 0.26 0.23 0.15 0.32 0.20 0.37 0.07 0.30 0.29 0.27
Coleslaw with Apple ... 0.06 0.27 0.11 0.08 0.36 1.00 0.29 0.06 0.22 0.45 0.31 0.21 0.28 0.43 0.43 0.37 0.07 0.45 0.37 0.31
Breakfast Burritos... 0.10 0.25 0.11 0.14 0.37 0.29 1.00 0.07 0.01 0.40 0.34 0.33 0.30 0.35 0.22 0.47 0.16 0.21 0.36 0.29
French 75... 0.33 0.14 0.08 0.31 0.02 0.06 0.07 1.00 0.04 0.13 0.07 0.04 0.11 0.06 0.10 0.06 0.21 0.06 0.07 0.06
Chipotle Cranberry S... 0.06 0.12 0.12 0.04 0.08 0.22 0.01 0.04 1.00 0.10 0.28 0.06 0.12 0.18 0.11 0.13 0.12 0.27 0.07 0.17
Warm Chanterelle Sal... 0.15 0.33 0.20 0.17 0.31 0.45 0.40 0.13 0.10 1.00 0.46 0.36 0.43 0.40 0.46 0.41 0.20 0.28 0.45 0.38
Rhubarb and Strawber... 0.17 0.32 0.40 0.19 0.26 0.31 0.34 0.07 0.28 0.46 1.00 0.38 0.30 0.36 0.38 0.35 0.21 0.26 0.43 0.36
Penne with Lemon and... 0.11 0.19 0.26 0.20 0.23 0.21 0.33 0.04 0.06 0.36 0.38 1.00 0.25 0.30 0.30 0.33 0.12 0.19 0.21 0.29
Marinated Mushroom, ... 0.15 0.48 0.19 0.19 0.15 0.28 0.30 0.11 0.12 0.43 0.30 0.25 1.00 0.34 0.39 0.30 0.29 0.27 0.41 0.39
Goat Cheese Toasts w... 0.14 0.28 0.19 0.13 0.32 0.43 0.35 0.06 0.18 0.40 0.36 0.30 0.34 1.00 0.34 0.40 0.07 0.28 0.37 0.41
Roasted Winter Veget... 0.19 0.38 0.22 0.11 0.20 0.43 0.22 0.10 0.11 0.46 0.38 0.30 0.39 0.34 1.00 0.40 0.22 0.20 0.31 0.24
Arugula and Roasted ... 0.13 0.29 0.18 0.24 0.37 0.37 0.47 0.06 0.13 0.41 0.35 0.33 0.30 0.40 0.40 1.00 0.13 0.22 0.37 0.22
Cocoa Delight... 0.20 0.36 0.11 0.16 0.07 0.07 0.16 0.21 0.12 0.20 0.21 0.12 0.29 0.07 0.22 0.13 1.00 0.20 0.15 0.28
Sauteed Collard Gree... 0.04 0.29 0.21 0.21 0.30 0.45 0.21 0.06 0.27 0.28 0.26 0.19 0.27 0.28 0.20 0.22 0.20 1.00 0.21 0.48
Pork, Vegetable, and... 0.10 0.39 0.30 0.08 0.29 0.37 0.36 0.07 0.07 0.45 0.43 0.21 0.41 0.37 0.31 0.37 0.15 0.21 1.00 0.44
Pumpkin-Pear Crisps... 0.05 0.41 0.28 0.11 0.27 0.31 0.29 0.06 0.17 0.38 0.36 0.29 0.39 0.41 0.24 0.22 0.28 0.48 0.44 1.00
1.0
0.8
0.6
0.4
0.2
Similarité Cosine

Matrice de similarité pour 20 recettes


Architecture du modèle, extraction d'embedding (pas d'entraînement)
Paramètres (pas d'entraînement)
Total: 4,049,571
Gelés (pré-entraînés)
0.0%
100.0%
Entraînables
Pipeline d'extraction d'embedding
Image 224x224x3
EfficientNet (Gelé)
Global Pool (1280-dim)
L2 Normalize
EXTRACTION DES EMBEDDINGS: 1x ce pipeline → Extraction de 1280-dim features
RECHERCHE: par Similarité Cosine entre l'input et la BDD des embeddings
Pas d'entraînement - extraction d'embeddings en utilisant les poids ImageNet
Statistiques de la base de données des embeddings
Count
14000
12000
10000
8000
6000
4000
2000
0
13,463
13,463
13,463
1,280
Total recettes
Avec images
Embeddings
Dim des embeddings
MODELE D'EXTRACTION D'EMBEDDING:
Architecture:
• Base: EfficientNetB0 (base pré-entraînée ImageNet)
• Output: 1280-dim (raw)
• Traitement: GlobalAveragePooling2D + L2 normalize
Paramètres:
• Total: 4,049,571
• Gelés: 4,049,571 (100% pré-entraînés)
• Entraînables: 0
Ce que fait ce modèle:
• Extraction d'embedding via EfficientNetB0 pré-entraîné
• GlobalAveragePooling2D → 1280 features
• L2 normalization
• Recherche par similarité Cosine
Ce que ça ne fait pas:
• Pas d'entraînement / transfer learning / fine-tuning
• Pas de custom Dense layers (tête custom)
• Pas d'optimisation spécifique aux recettes
Caractéristiques:
• Strategie: recherche de similarité Cosine
• Forces: reconnaissance correcte mais limité des images de recettes similaires
• Limites: ne comprend pas forcément toutes les subtilités propres aux recettes
• Vitesse: très rapide (pas d'entraînement), on extrait juste les embeddings
Statut: Prêt pour la recherche de similarité
Dataset: 13,463 recettes, 13,463 avec images

10. Sauvegarde complète du système

In [11]:
```python
import pickle

def save_complete_system(retrieval_system):
    """Save the complete retrieval system"""
    print(" Saving complete retrieval system...")

    # Sauvegarde du modèle
    retrieval_system.model.save("./raw/embeddings_recipe_image_retrieval_model_raw.keras")
    print("Modèle sauvegardé: ./raw/embeddings_recipe_image_retrieval_model_raw.keras")

    # Sauvegarde de la base de données des embeddings
    database_path = "./raw/recipe_embeddings_database_raw"
    metadata_path = "./raw/recipe_embeddings_database_metadata_raw"
    np.save(f"{database_path}.npy", retrieval_system.embeddings_db)
    print(f"Embeddings sauvegardés: {database_path}.npy ({retrieval_system.embeddings_db.shape})")

    # Sauvegarde des métadonnées (en utilisant les attributs actuels)
    metadata = {
        'image_paths': retrieval_system.image_paths,
        'image_to_recipe_map': retrieval_system.image_to_recipe_map
    }
    with open(f"{metadata_path}.pkl", 'wb') as f:
        pickle.dump(metadata, f)
    print(f"Metadata sauvegardées: ./raw/recipe_embeddings_database_metadata_raw.pkl")

    # Sauvegarde du DataFrame 'recipes_with_images'
    recipes_df_path = "./data/recipes_with_images_dataframe.pkl"
    retrieval_system.recipes_df.to_pickle(recipes_df_path)
    print(f"DataFrame 'recipes_with_images' sauvegardé: {recipes_df_path}")

    model_size = os.path.getsize("./raw/embeddings_recipe_image_retrieval_model_raw.keras") / (1024**2)
    embeddings_size = os.path.getsize(f"{database_path}.npy") / (1024**2)
    metadata_size = os.path.getsize(f"{metadata_path}.pkl") / (1024**2)
    df_size = os.path.getsize(recipes_df_path) / (1024**2)

    print(f"\nTaille des fichiers:")
```

```
    print(f"Model: {model_size:.1f} MB")
    print(f"Embeddings: {embeddings_size:.1f} MB")
    print(f"Metadata: {metadata_size:.1f} MB")
    print(f"    DataFrame: {df_size:.1f} MB")

    print(f"\nSauvegarde complète du modèle et des {len(retrieval_system.embeddings_db):,} embeddings effectuée!")

save_complete_system(retrieval_system)
```

```
 Saving complete retrieval system...
Modèle sauvegardé: ./raw/embeddings_recipe_image_retrieval_model_raw.keras
Embeddings sauvegardés: ./raw/recipe_embeddings_database_raw.npy ((13463, 1280))
Metadata sauvegardées: ./raw/recipe_embeddings_database_metadata_raw.pkl
DataFrame 'recipes_with_images' sauvegardé: ./data/recipes_with_images_dataframe.pkl

Taille des fichiers:
Model: 16.2 MB
Embeddings: 65.7 MB
Metadata: 22.2 MB
    DataFrame: 73.2 MB

Sauvegarde complète du modèle et des 13,463 embeddings effectuée!
```

11. Test du recherche avec une image aléatoire du dataset

In [12]:
```
# Test avec une image aléatoire du dataset
# Si possible, on recharge le système depuis les fichiers sauvegardés => parfait pour préparer le déploiement

import tensorflow as tf
import pickle
import numpy as np
import os
import pandas as pd

def l2_norm(x):
    return tf.nn.l2_normalize(x, axis=1)

def same_shape(x):
    return x
```

```
CONFIG_STANDALONE = {
    'IMG_SIZE': 224,
}

def ensure_retrieval_system():
    """Init du système de recherche avec chargement depuis les fichiers sauvegardés si possible"""
    global retrieval_system

    # On vérifie si le système existe déjà et est prêt
    if 'retrieval_system' in globals() and retrieval_system is not None and retrieval_system.embeddings_db is not None:
        print("Utilisation du système existant en mémoire")
        return True

    print("Chargement du système depuis les fichiers sauvegardés...")

    try:

        # On a besoin des custom l2_norm et same_shape pour le chargement du modèle
        custom_objects = {
            'l2_norm': l2_norm,
            'same_shape': same_shape,
        }

        # Chemin du modèle sauvegardé précédemment
        model_paths = [
            "./raw/embeddings_recipe_image_retrieval_model_raw.keras"
        ]

        model = None
        for model_path in model_paths:
            if os.path.exists(model_path):
                try:
                    model = tf.keras.models.load_model(model_path,
                                                       compile=False,
                                                       safe_mode=False,
                                                       custom_objects=custom_objects)
                    print(f"Modèle chargé depuis {model_path}")
                    break
                except Exception as e:
                    print(f"Echec de chargement depuis {model_path}: {e}")
                    continue
```

```
    if model is None:
        print(" Could not load model from any location")
        return False

    # Chargement de la base de données des embeddings
    embeddings_db = np.load("./raw/recipe_embeddings_database_raw.npy")
    print(f"Embeddings chargés: {embeddings_db.shape}")

    # Chargement des métadonnées
    with open("./raw/recipe_embeddings_database_metadata_raw.pkl", 'rb') as f:
        metadata = pickle.load(f)
    print("Métadonnées chargées")

    # Chargement du DataFrame 'recipes_with_images'
    recipes_df_path = "./data/recipes_with_images_dataframe.pkl"
    if os.path.exists(recipes_df_path):
        recipes_with_images = pd.read_pickle(recipes_df_path)
        print(f"DataFrame chargée: {len(recipes_with_images)} recipes")
    else:
        print("Fichier recipes_with_images_dataframe.pkl introuvable!")
        print("Il faut d'abord sauvegarder le DataFrame 'recipes_with_images' dans les cellules précédentes")
        return False

    # On crée le système avec les différents éléments chargés
    retrieval_system = RecipeImageRetrieval(model, recipes_with_images)
    retrieval_system.embeddings_db = embeddings_db
    retrieval_system.image_paths = metadata['image_paths']
    retrieval_system.image_to_recipe_map = metadata['image_to_recipe_map']

    print(f"Système chargé avec {len(embeddings_db):,} embeddings!")
    return True

except FileNotFoundError as e:
    print(f"Fichier introuvable: {e}")
    print("Vérifiez que ces fichiers existent:")
    print("- ./raw/embeddings_recipe_image_retrieval_model_raw.keras")
    print("- ./raw/recipe_embeddings_database_raw.npy")
    print("- ./raw/recipe_embeddings_database_metadata_raw.pkl")
    print("- ./data/recipes_with_images_dataframe_raw.pkl")
    return False
```

```
    except Exception as e:
        print(f"Erreur lors du chargement du système: {e}")
        import traceback
        traceback.print_exc()
        return False

# Main testing logic
if ensure_retrieval_system():
    print("\nTest avec une image aléatoire du dataset...")
    retrieval_system.test_with_random_image(top_k=3)
else:
    print("\nImpossible d'initialiser le système")
```

```
Utilisation du système existant en mémoire

Test avec une image aléatoire du dataset...
Test avec une image aléatoire: C:\Users\Naturel\.cache\kagglehub\datasets\pes12017000148\food-ingredients-and-recipe-dataset-with-images\versions\1\Food Images\Food Images\arugula-and-ricotta-calzones-233791.jpg
Recherche de 3 recettes similaires...
3 recipes similaires trouvées
```

```
================================================================================
RECETTES SIMILAIRES
================================================================================

RANG #1: 1.0000
Arugula and Ricotta Calzones
------------------------------------------------------------
Ingredients:
['1 large garlic clove, minced', '2 tablespoons extra-virgin olive oil', '5 oz baby arugula (8 cups packed)', '6 oz whole-milk
ricotta (2/3 cup)', '3 oz whole-milk mozzarella, coarsely grated', '2 tablespoonsfinely grated Parmigiano-Reggiano', '1 large e
gg yolk', '1/4 teaspoon salt', '1/8 teaspoon b...
------------------------------------------------------------
Instructions:
Put oven rack in lower third of oven and preheat oven to 450°F.
Cook garlic in oil in a 12-inch heavy skillet over moderate heat, stirring frequently, until golden, 1 to 2 minutes. Add arugul
a and cook, stirring frequently, until wilted, 2 to 3 minutes. Transfer to a sieve and press hard on arugula to squeeze out as
much excess liquid as possible, then coarsely chop.
Stir together ricotta, mozzare...
================================================================================

RANG #2: 0.6778
Lemon-Rhubarb Chicken
------------------------------------------------------------
Ingredients:
['5 tablespoons olive oil, divided', '2 tablespoons plus 1/4 cup chopped shallots', '4 1/2 cups diced rhubarb, divided', '1 tab
lespoon fresh lemon juice', '2 teaspoons finely grated lemon peel, divided', '1/4 cup (1/2 stick) butter', '1/2 cup sliced unpe
eled fresh ginger', '3/4 cup sugar', '6 tables...
------------------------------------------------------------
Instructions:
Heat 2 tablespoons olive oil in heavy large skillet over medium-high heat. Add 2 tablespoons chopped shallots and 2 cups rhubar
b; sauté until soft, about 5 minutes. Stir in lemon juice and 1 teaspoon lemon peel. Season with salt and pepper. Cool rhubarb
stuffing.
Melt butter in heavy large saucepan over low heat. Add 21/2 cups rhubarb, 1/4 cup shallots, and ginger; sauté until soft, about
10 minut...
================================================================================

RANG #3: 0.6708
Shrimp and Cotija Enchiladas with Salsa Verde and Crema Mexicana
------------------------------------------------------------
Ingredients:
['4 tablespoons olive oil, divided', '2 1/2 pounds tomatillos,* husked, rinsed', '4 large fresh poblano chiles,* halved lengthw
```

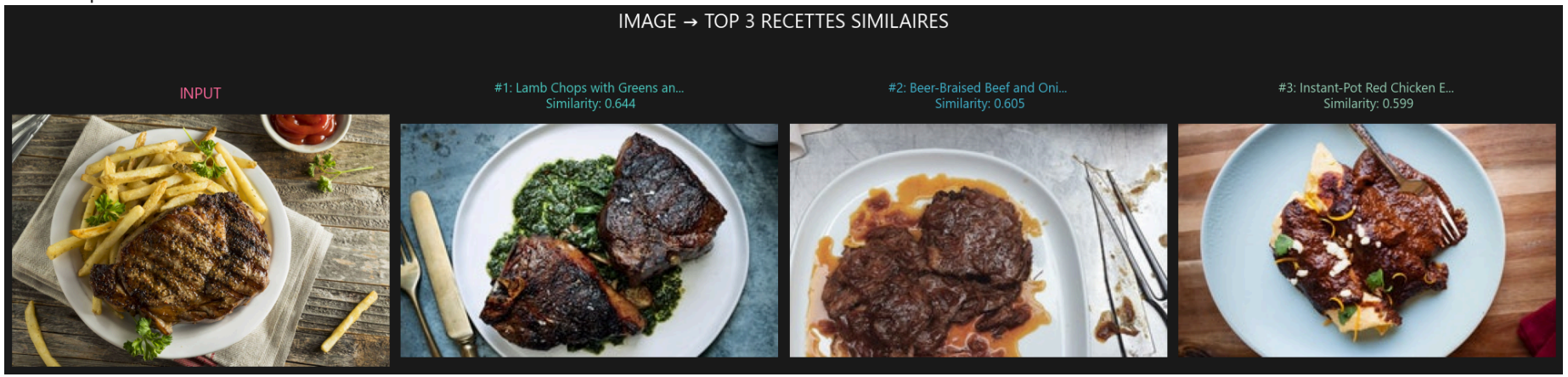
```
ise, cored, seeded', '4 unpeeled garlic cloves', '2 cups (packed) coarsely chopped fresh cilantro plus additional for garnish',
'plus additional for garnish', '1 cup (packed...
------------------------------------------------------------
Instructions:
Preheat broiler. Line large rimmed baking sheet with foil; brush lightly with 1 tablespoon olive oil. Arrange tomatillos, pobla
no chiles, cut side down, and garlic cloves on prepared baking sheet. Broil until tomatillos and chiles begin to soften and bla
cken in several spots, watching closely to prevent burning, about 10 minutes (do not turn). Remove from oven. Let stand until c
ool enough to handl...
================================================================================
```

12. Recherches sur des images inconnues du système

In [13]:
```python
import tensorflow as tf
import numpy as np

if 'retrieval_system' not in globals() or retrieval_system.embeddings_db is None:
    print("Chargement du système sauvegardé...")

    try:
       # Chargement du modèle
        model = tf.keras.models.load_model("./raw/recipe_image_retrieval_model_raw.keras")
        embeddings_db = np.load("./raw/recipe_embeddings_database_raw.npy")

        import pickle
        with open("./raw/recipe_embeddings_database_metadata_raw.pkl", 'rb') as f:
            metadata = pickle.load(f)

        # On crée le système avec les différents éléments chargés
        retrieval_system = RecipeImageRetrieval(model, recipes_with_images)
        retrieval_system.embeddings_db = embeddings_db
        retrieval_system.image_paths = metadata['image_paths']
        retrieval_system.image_to_recipe_map = metadata['image_to_recipe_map']

    except FileNotFoundError:
        print("Aucun système sauvegardé trouvé.")
        retrieval_system = None
    except Exception as e:
        print(f"Erreur lors du chargement: {e}")
        retrieval_system = None
```

```python
if retrieval_system and retrieval_system.embeddings_db is not None:
    # Test sur les images 1.jpg à 8.jpg
    import os
    for i in range(1, 9):
        test_image = f"./test_recipes/{i}.jpg"

        if os.path.exists(test_image):
            print(f"\n{'='*50}")
            print(f"TEST IMAGE {i}.jpg")
            print(f"{'='*50}")

            results = retrieval_system.search_similar_recipes(test_image, top_k=3)
            if results:
                retrieval_system.display_results(test_image, results)
            else:
                print("Aucun résultat trouvé")
        else:
            print(f"Image {i}.jpg introuvable")

else:
    print("Système non disponible")
```

```
==================================================
TEST IMAGE 1.jpg
==================================================
Recherche de 3 recettes similaires...
3 recipes similaires trouvées
```



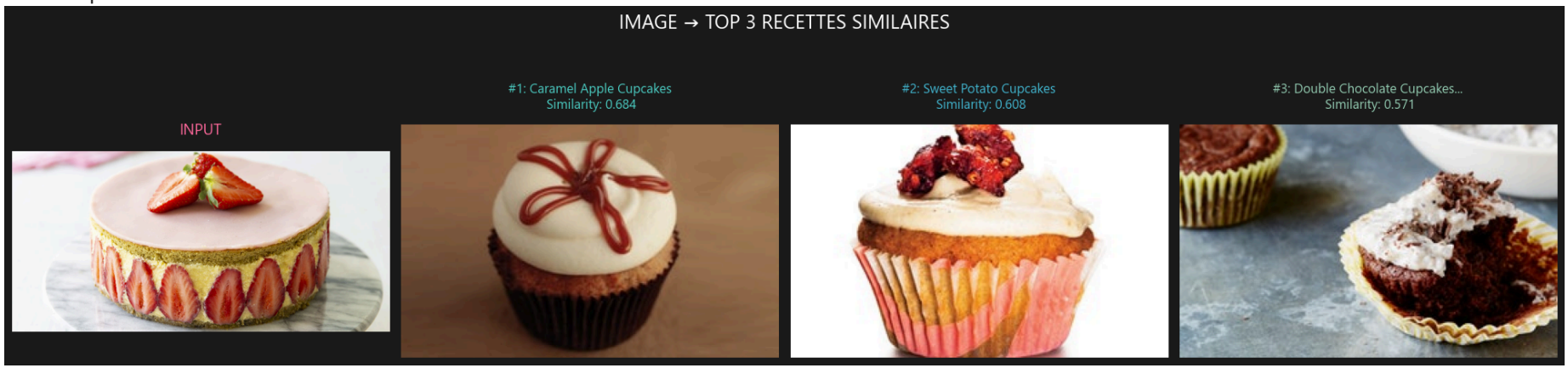
```
================================================================================
RECETTES SIMILAIRES
================================================================================

RANG #1: 0.6437
Lamb Chops with Greens and Sorrel Salsa Verde
------------------------------------------------------------
Ingredients:
['8 1 1/2-2"-thick lamb loin chops (about 3 pounds)', 'Kosher salt, freshly ground pepper', '2/3 cup sorrel or spinach leaves',
'1/3 cup mint leaves', '1/3 cup parsley leaves', '1 tablespoon finely grated lemon zest', '1 teaspoon crushed red pepper flake
s', '5 garlic cloves, thinly sliced, divided',...
------------------------------------------------------------
Instructions:
Pat lamb dry; season with salt and pepper. Let sit at room temperature 1 hour.
Meanwhile, puree sorrel, mint, parsley, lemon zest, red pepper flakes, about 1 garlic clove, and 1/3 cup oil in a blender, scra
ping down sides as needed, until mixture is smooth. Season salsa verde with salt and pepper; set aside.
Prepare grill for high heat (or heat a grill pan over high). Grill chops, turning every 2 ...
================================================================================

RANG #2: 0.6049
Beer-Braised Beef and Onions
------------------------------------------------------------
Ingredients:
['3 pounds onions', '1 (5-pound) boneless beef chuck roast, tied', '2 tablespoons vegetable oil, divided', '2 Turkish bay leave
s or 1 California', '2 (12-ounces) bottles pilsner-style beer such as Budweiser', '2 tablespoons red-wine vinegar']
------------------------------------------------------------
Instructions:
Halve onions lengthwise, then slice lengthwise 1/4 inch thick.
Pat beef dry and season all over with 2 1/2 teaspoons salt and 1 teaspoon pepper. Heat 1 tablespoon oil in a wide 5-to 6-quart
heavy pot over medium-high heat until it shimmers. Brown beef on all sides, about 15 minutes, then transfer to a plate.
Cook onions with bay leaves and 1/2 teaspoon salt in remaining tablespoon oil in pot, scra...
================================================================================

RANG #3: 0.5989
Instant-Pot Red Chicken Enchiladas
------------------------------------------------------------
Ingredients:
['1 pound boneless, skinless chicken thighs', '2 cups Ancho Chile Sauce', '1/2 cup chopped onion', '12 corn tortillas', '3 tabl
espoons vegetable oil or nonstick cooking spray', '1/2 cup crumbled queso fresco', '2 cups shredded Monterey Jack cheese']
------------------------------------------------------------
Instructions:
```

Put the chicken thighs in the Instant Pot. Pour in the ancho chile sauce. Lock the lid into place. Select Manual; adjust the pr
essure to High and the time to 10 minutes. After cooking, naturally release the pressure for 10 minutes, then quick release any
remaining pressure. Unlock and remove the lid. Transfer the chicken to a bowl and let it cool for a few minutes. Keep the sauce
warm.
When the ch...
==

==
TEST IMAGE 2.jpg
==
Recherche de 3 recettes similaires...
3 recipes similaires trouvées

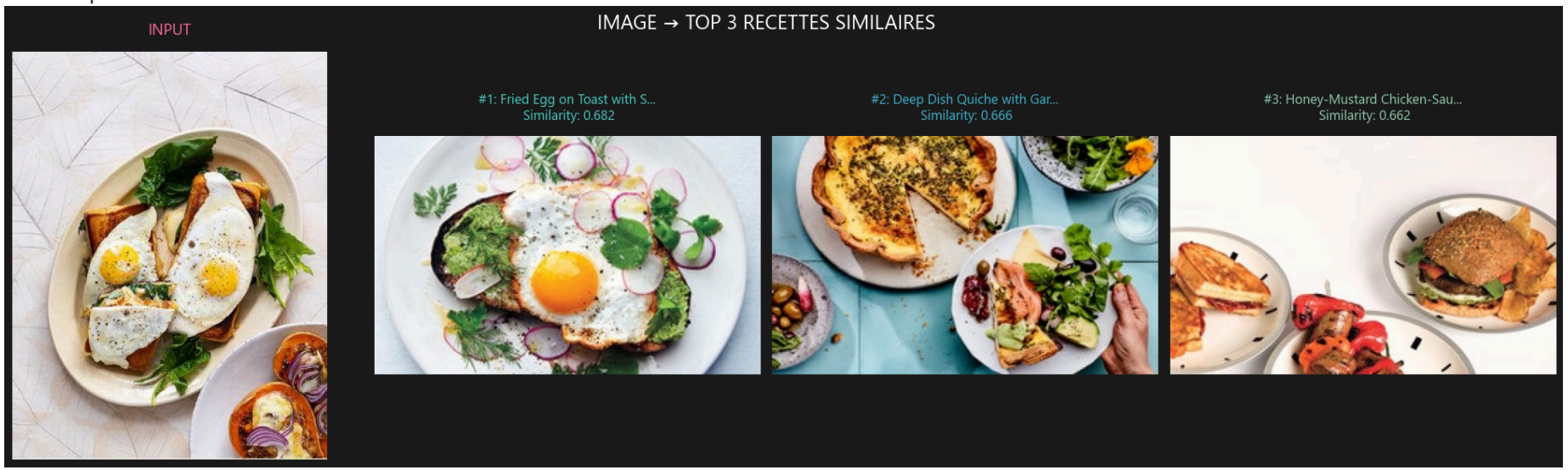
```
================================================================================
RECETTES SIMILAIRES
================================================================================

RANG #1: 0.5560
Glazed Cinnamon-Cardamom Buns
------------------------------------------------------------
Ingredients:
['1 cup whole milk', '1 Tbsp. active dry yeast', '1 large egg', '1 large egg yolk', '3 1/2 cups (475 g) all-purpose flour', '1/
2 cup (105 g) granulated sugar', '1 1/2 tsp. (3 g) ground cardamom', '1 tsp. kosher salt', '6 Tbsp. room temperature unsalted b
utter, plus more for bowl', '6 Tbsp. unsalted ...
------------------------------------------------------------
Instructions:
Heat milk in a small saucepan over low until just warm; an instant-read thermometer should register 105°F–115°F. Pour into the
large bowl of a stand mixer. Whisk in yeast and let sit until foamy, 10–15 minutes. You should see a layer of foam on the surfa
ce; this means that the yeast is active.
Add egg, egg yolk, flour, granulated sugar, cardamom, and salt to yeast mixture and mix with dough hook o...
================================================================================

RANG #2: 0.5442
Crunchy Chili Onion Rings
------------------------------------------------------------
Ingredients:
['3 cups all purpose flour', '2 tablespoons chili powder', '4 1/2 teaspoons salt', '1 12-ounce bottle dark beer (preferably Mex
ican)', '12 6-inch corn tortillas, coarsely torn', '2 large onions, peeled', 'Vegetable oil (for deep-frying)']
------------------------------------------------------------
Instructions:
Mix flour, chili powder, and salt in medium bowl. Pour beer into small bowl. Finely grind tortillas in processor; transfer to d
eep bowl.
Line large baking sheet with foil. Cut onions crosswise into 1/2- to 3/4-inch-thick rounds. Separate rounds into rings. Dip 1 o
nion ring into flour mixture, then beer, flour again and beer again, then add to bowl with ground tortillas and toss to coat. U
sing wood...
================================================================================

RANG #3: 0.4912
Plain Bagels
------------------------------------------------------------
Ingredients:
['1 envelope active dry yeast (1/4 ounce)', '1 1/2 tablespoons plus 1/4 cup honey, divided', '1 tablespoon non-diasatic malt po
wder (such as King Arthur brand)', '3 tablespoons neutral oil, plus more for greasing', '9 cups high gluten flour such as bread
flour', '2 teaspoons kosher salt']
```

```
------------------------------------------------------------
Instructions:
In the bowl of a stand mixer, whisk together 2 2/3 cups water, yeast, and 1 1/2 tablespoons honey. Let stand for 5 minutes.
Add malt, oil, flour, and salt and mix with the dough hook attachment at low speed for about 10 minutes, or until dough can "pu
ll a window." To test, pinch off a small ball of dough and pull into a thin, see-through membrane without it tearing. If it tea
rs, mix another minute...
================================================================================


==================================================
TEST IMAGE 3.jpg
==================================================
Recherche de 3 recettes similaires...
3 recipes similaires trouvées
```



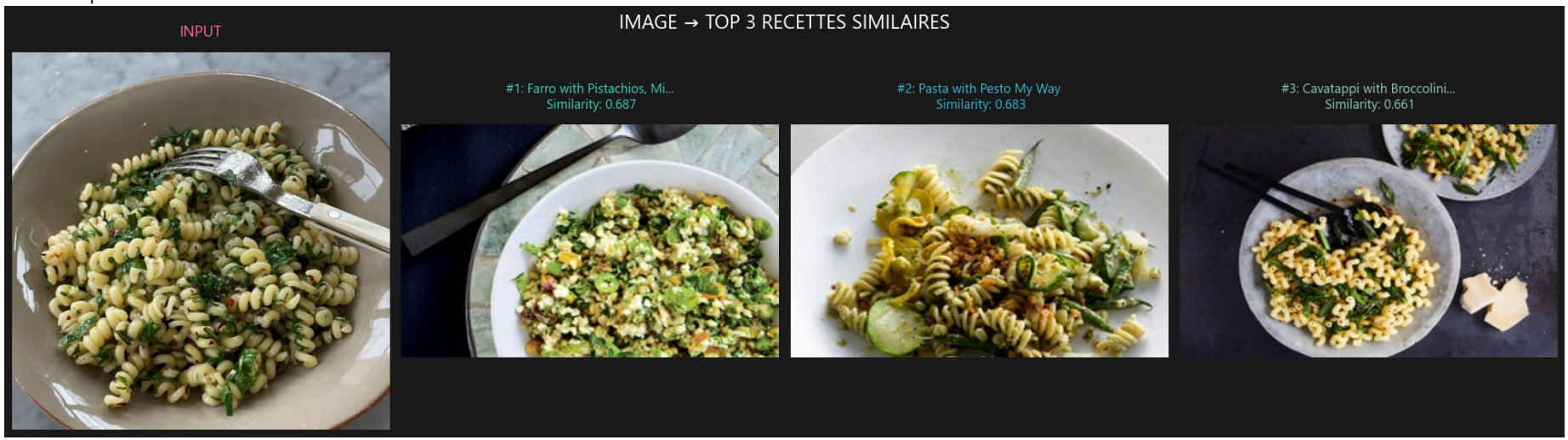
```
================================================================================
RECETTES SIMILAIRES
================================================================================

RANG #1: 0.6839
Caramel Apple Cupcakes
------------------------------------------------------------
Ingredients:
['2 1/2 cups all-purpose flour', '3 tsp. baking powder', '1 tsp. ground cinnamon', '1/2 tsp. salt', '16 Tbsp. unsalted butter,
at room temperature (European style recommended)', '2 cups sugar', '4 large eggs, at room temperature', '1/3 cup hot water', '2
1/2 cups freshly grated apples (4 to 5 medium...
------------------------------------------------------------
Instructions:
Preheat the oven to 350°F. Line a standard cupcake pan with twelve paper baking cups, and a second pan with six baking cups, or
grease pans with butter if not using baking cups.
Sift together the flour, baking powder, cinnamon, and salt on a sheet of parchment paper or wax paper and set aside.
Place the butter in the bowl of a stand mixer or in a bowl with a handheld electric mixer. Beat on medium...
================================================================================

RANG #2: 0.6076
Sweet Potato Cupcakes
------------------------------------------------------------
Ingredients:
['1/2 cup oat flour', '6 tablespoons all-purpose flour', '1/2 teaspoon baking soda', '6 tablespoons granulated sugar', '2 table
spoons unsalted butter, at room temperature', '1 egg white', '1/4 cup cooked or canned sweet potato', '1/4 teaspoon vanilla ext
ract', '2 tablespoons skim milk', "1/4 cup con...
------------------------------------------------------------
Instructions:
Heat oven to 350°F. Line 16 cups of two 12-cup mini-muffin pans with mini-muffin papers. In a bowl, combine flours and baking s
oda. In another bowl, cream together sugar and butter with an electric mixer on medium. Add egg white, sweet potato and vanill
a; beat on low until well combined. Add flour mixture and milk; beat on low until just combined. Do not overmix. Fill muffin cu
ps 2/3 full. Bake un...
================================================================================

RANG #3: 0.5712
Double Chocolate Cupcakes With Salted Chia Pudding Frosting
------------------------------------------------------------
Ingredients:
['1 (14-ounce) can full-fat coconut milk', '5 Medjool dates, pitted and chopped', '1/4–1/2 teaspoon of sea salt', '5 tablespoon
s chia seeds', '1 cup unsweetened almond milk', '1 tablespoon apple cider vinegar', '1 tablespoon flax meal', '3 tablespoons wa
ter', '1 cup chickpea (garbanzo bean) flour', ...
```

```
------------------------------------------------------------
Instructions:
Combine the coconut milk, dates, and salt in a blender and blend until the mixture is smooth. This might take a few minutes. If
end up with small bits of dates, that's OK. But you don't want to see chunks or even white slivers of the coconut milk. The col
or should be a very light tan. Taste, and if you want to add more salt, do it now and then blend for a few more seconds (I real
ly like the saltin...
================================================================================

==================================================
TEST IMAGE 4.jpg
==================================================
Recherche de 3 recettes similaires...
3 recipes similaires trouvées
```



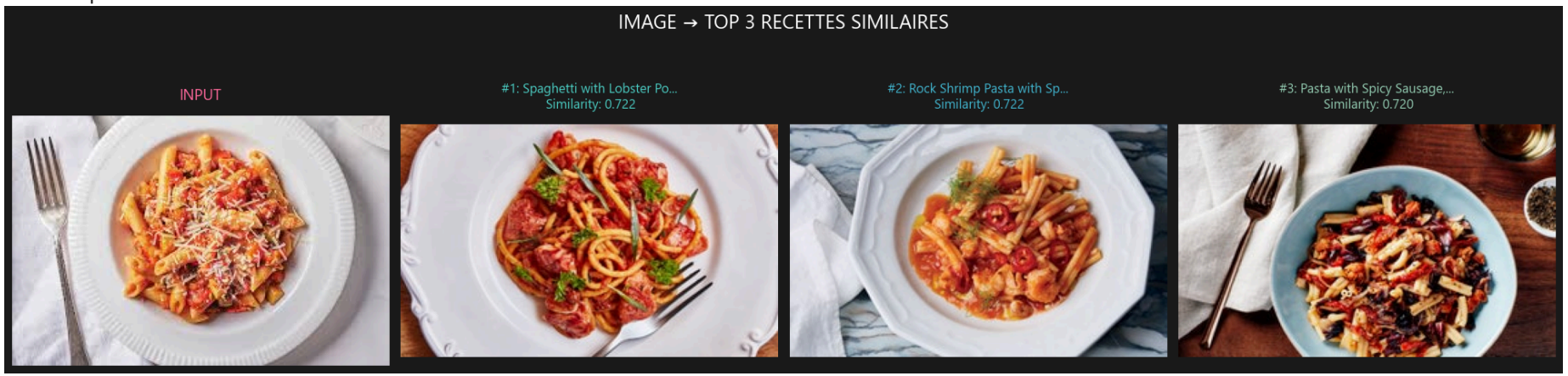
```
================================================================================
RECETTES SIMILAIRES
================================================================================

RANG #1: 0.6819
Fried Egg on Toast with Salted Herb Butter and Radishes
------------------------------------------------------------
Ingredients:
['3/4 cup mixed tender herbs (such as parsley, dill, basil, chives, and/or tarragon), plus sprigs for serving', '1/2 cup (1 sti
ck) unsalted butter, room temperature, cut into pieces', '1/2 teaspoon finely grated lemon zest', '2 teaspoons fresh lemon juic
e, plus more for drizzling', '1 teaspoon koshe...
------------------------------------------------------------
Instructions:
Pulse 3/4 cup herbs in a food processor until finely chopped. Add butter, lemon zest, 2 tsp. lemon juice, 1 tsp. salt, and 1/2
tsp. pepper. Pulse to bring together.
Place radishes in a medium bowl; drizzle with a little lemon juice and add a pinch of salt. Toss to combine.
To serve, spread toast with herb butter and top each with a fried egg. Scatter dressed radishes over top along with a few spr
i...
================================================================================

RANG #2: 0.6660
Deep Dish Quiche with Garnishes
------------------------------------------------------------
Ingredients:
['1 full recipe Our Favorite Pie Dough', 'All-purpose flour (for surface)', '2 shallots, finely chopped', '2 Tbsp. extra-virgin
olive oil', '1 1/2 tsp. kosher salt, divided', '1 Tbsp. finely chopped chives', '8 large eggs', '1 3/4 cups half-and-half', '1/
2 tsp. freshly ground black pepper', '1 cup c...
------------------------------------------------------------
Instructions:
Roll dough to a 15" round on a lightly floured work surface; trim edges. Fit dough into a 9" springform pan, letting dough fold
over itself slightly around the edges so that it creates a gently pleated, rippled effect (the dough will extend above top of p
an). Place pan on a rimmed baking sheet. Chill 1 hour, or freeze 20 minutes.
Preheat oven to 350°F. Dock bottom surface of dough with a fork. Lin...
================================================================================

RANG #3: 0.6623
Honey-Mustard Chicken-Sausage Kebabs
------------------------------------------------------------
Ingredients:
['6 tablespoons Dijon mustard', '2 tablespoons honey', '1 tablespoon mayonnaise', '4 fully cooked sun-dried-tomato chicken saus
ages, each cut into 6 pieces', '24 mini bell peppers', 'Olive oil', 'Metal skewers']
```

```
------------------------------------------------------------
Instructions:
Prepare barbecue (medium-high heat). Whisk mustard, honey, and mayonnaise in small bowl to blend. Thread 3 sausage pieces alter
natively with 3 peppers onto each of 8 skewers and place on baking sheet. Brush with milk; sprinkle with salt and pepper.
Grill skewers until vegetables are lightly charred and crisp-tender and sausage is heated through, turning occasionally and bru
shing with mustard mixtu...
================================================================================

==================================================
TEST IMAGE 5.jpg
==================================================
Recherche de 3 recettes similaires...
3 recipes similaires trouvées
```



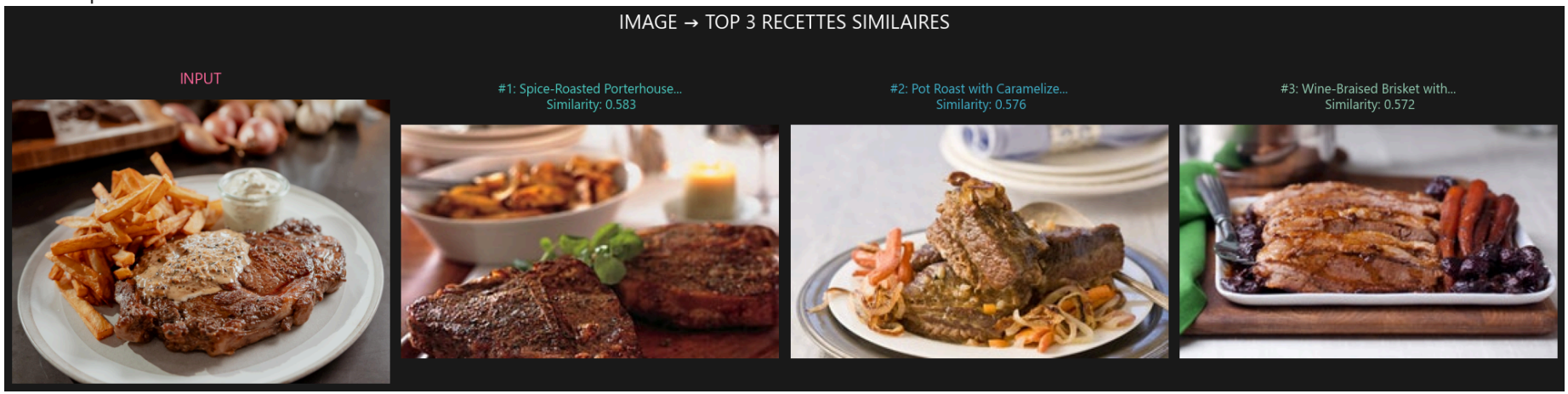
```
================================================================================
RECETTES SIMILAIRES
================================================================================

RANG #1: 0.6865
Farro with Pistachios, Mixed Herbs, and Golden Raisins
------------------------------------------------------------
Ingredients:
['2 cups farro', '1/2 teaspoon kosher salt, plus more', '1/2 cup pistachios', '1 1/2 teaspoons finely grated lemon zest', '3 ta
blespoons fresh lemon juice', '1 teaspoon finely grated peeled ginger', '1/2 teaspoon sugar', '1/3 cup grapeseed oil or olive o
il', 'Freshly ground black pepper', '1 serrano...
------------------------------------------------------------
Instructions:
Preheat oven to 350°F. Rinse farro under cold water. Cook in a large pot of boiling salted water, skimming surface occasionall
y, until tender, 20–25 minutes.
Meanwhile, toast pistachios on a rimmed baking sheet, tossing once, until golden brown, 8–10 minutes. Let cool, then coarsely c
hop.
Whisk lemon zest, lemon juice, ginger, sugar, and 1/2 tsp. salt in a medium bowl. Whisking constantly, gradual...
================================================================================

RANG #2: 0.6827
Pasta with Pesto My Way
------------------------------------------------------------
Ingredients:
['2 medium zucchini (about 1/2 pound total)', '1/2 pound boiling potatoes, peeled', '3/4 pound rotini or penne', '6 ounces gree
n beans, trimmed and halved (about 1 1/2 cups)', '2 ears corn, kernels cut off (about 1 1/2 cups)', '1 tablespoon olive oil',
'1 pint cherry tomatoes (12 to 14 ounces)', "1 ...
------------------------------------------------------------
Instructions:
Cut zucchini into thin rounds with slicer. Fit slicer with julienne attachment and cut potatoes into 1/4-inch-thick matchstick
s.
Start cooking rotini in a pasta pot of boiling salted water (3 tablespoons salt for 6 quarts water) according to package instru
ctions. Six minutes before pasta is done, stir in potatoes and green beans. Two minutes before pasta is done, stir in corn and
zucchini.
Meanwhi...
================================================================================

RANG #3: 0.6611
Cavatappi with Broccolini, Brown Butter, and Sage
------------------------------------------------------------
Ingredients:
```

```
['Kosher salt', '2 bunches Broccolini (about 1 pound), ends trimmed, split lengthwise into halves or thirds depending on the th
ickness (or substitute broccoli rabe or broccoli)', '2 tablespoons extra-virgin olive oil', '2 large cloves garlic, minced',
'1/4 teaspoon crushed red pepper flakes', 'Fresh...
------------------------------------------------------------
Instructions:
Bring a large pot of water to a boil. Fill a large bowl with water and ice and set aside.
Add 1 tablespoon kosher salt and the broccolini to the boiling water and cook until crisp-tender, 2 to 3 minutes. Using a spide
r or slotted spoon, transfer the Broccolini to the ice water to stop the cooking and let cool. Keep the pot of water boiling fo
r the pasta. Drain the broccolini in a colander, cut the...
================================================================================

==================================================
TEST IMAGE 6.jpg
==================================================
Recherche de 3 recettes similaires...
3 recipes similaires trouvées
```



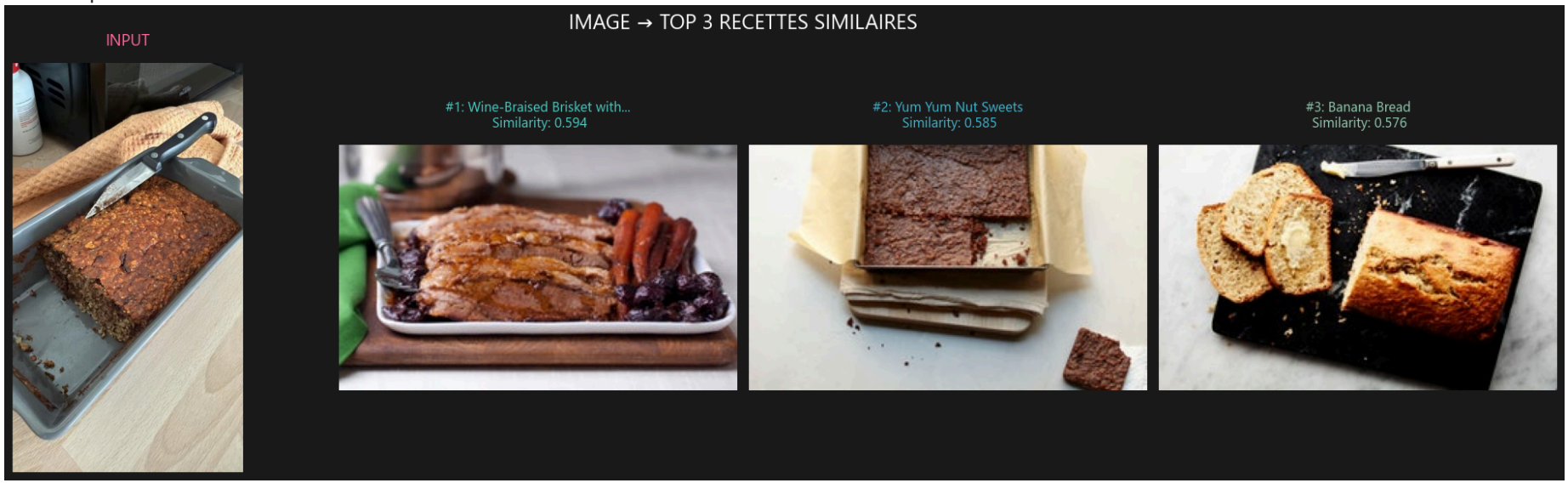
```
================================================================================
RECETTES SIMILAIRES
================================================================================

RANG #1: 0.7219
Spaghetti with Lobster Pomodoro
------------------------------------------------------------
Ingredients:
['Kosher salt', '2 (1 1/4-lb.) live lobsters', '2 Tbsp. extra-virgin olive oil', '2 Tbsp. ghee or unsalted butter', '1/4 small
red onion, thinly sliced', '3 garlic cloves, thinly sliced', '2 sprigs basil', '1 (14-oz.) can whole peeled San Marzano tomatoe
s', '12 oz. spaghetti', '4 oz. nduja, broken i...
------------------------------------------------------------
Instructions:
Bring a large pot of salted water to a rolling boil. Working one at a time, cook lobsters 3 minutes, then transfer to a large b
owl of ice water. Let cool just until you can comfortably handle them, about 1 minute, then twist off claws where the knuckles
meet the body and return them to pot of boiling water. Cook 2 minutes (leave bodies in ice water). Add claws back to ice water
and let both claws ...
================================================================================

RANG #2: 0.7216
Rock Shrimp Pasta with Spicy Tomato Sauce
------------------------------------------------------------
Ingredients:
['1 (28-ounce) can whole peeled tomatoes, preferably San Marzano, drained', '1/3 cup olive oil, plus more for drizzling', '1/2
medium fennel bulb, fronds reserved, core removed, bulb thinly sliced', '8 garlic cloves, smashed', '1 Fresno chile, very thinl
y sliced, divided', '1/4 cup dry white wine', ...
------------------------------------------------------------
Instructions:
Set a fine-mesh sieve over a medium bowl. Working over sieve, squeeze tomatoes to release juices and break up flesh. Let tomato
es drain in sieve, collecting juices in bowl, until ready to use.
Heat 1/3 cup oil in a large Dutch oven or other heavy pot over medium. Cook fennel, garlic, and half of chile, stirring often,
until garlic is golden and fennel is starting to brown around the edges, 5–8 min...
================================================================================

RANG #3: 0.7201
Pasta with Spicy Sausage, Radicchio, and Sun-Dried Tomatoes
------------------------------------------------------------
Ingredients:
['1 pound casarecce or other short pasta', 'Kosher salt', '1/4 cup olive oil', '1 large onion, thinly sliced', '1 pound spicy I
talian sausage, casings removed', '1 1/4 pounds radicchio, cored, coarsely chopped (about 7 cups)', '1/2 cup thinly sliced drai
ned oil-packed sun-dried tomatoes', 'Freshly g...
```

```
------------------------------------------------------------
Instructions:
Cook pasta in a large pot of boiling salted water, stirring occasionally, until al dente. Drain pasta, reserving 3/4 cup pasta
cooking liquid; return pasta to pot.
Meanwhile, heat oil in a medium skillet over medium-high. Add onion and 1/4 tsp. salt and cook, stirring occasionally, until te
nder, about 5 minutes (do not brown). Add sausage and cook, stirring occasionally and breaking up sausage int...
================================================================================

==================================================
TEST IMAGE 7.jpg
==================================================
Recherche de 3 recettes similaires...
3 recipes similaires trouvées
```



```
================================================================================
RECETTES SIMILAIRES
================================================================================

RANG #1: 0.5833
Spice-Roasted Porterhouse Steaks
------------------------------------------------------------
Ingredients:
['Two 2-inch-thick porterhouse steaks', 'Olive oil', '1 1/2 tablespoons Coriander-Herb Spice Rub']
------------------------------------------------------------
Instructions:
Place two 2-inch-thick porterhouse steaks on a heavy rimmed baking sheet. Brush both sides with a full-flavored olive oil, then
rub 1 1/2 tablespoons of the Coriander-Herb Spice Rub into each side of each steak.
Preheat the oven to 450°F. Roast steaks until a meat thermometer inserted into the center registers 125°F for rare, about 25 mi
nutes. Transfer the steaks to a cutting board, cover loosely ...
================================================================================

RANG #2: 0.5763
Pot Roast with Caramelized Onions and Roasted Carrots
------------------------------------------------------------
Ingredients:
['1/2 cup canola oil', 'Kosher salt and freshly ground black pepper', '5 pounds boneless short ribs, denuded (all surface fat r
emoved; have your butcher do this)', '1 cup dry sherry', '4 carrots, peeled and roughly chopped', '2 large onions, peeled and r
oughly chopped', '8 stalks celery, peeled and ...
------------------------------------------------------------
Instructions:
Position racks in upper and lower thirds of oven and preheat to 350°F. Season beef liberally with salt and pepper. In large Dut
ch oven or heavy ovenproof pot over moderately high heat, heat oil until hot but not smoking. Add beef and sear until dark brow
n and crisp on both sides, about 10 minutes total. Transfer beef to large plate. Pour off oil in pan and discard. Add sherry an
d simmer uncovered,...
================================================================================

RANG #3: 0.5723
Wine-Braised Brisket with Tart Cherries
------------------------------------------------------------
Ingredients:
["1/4 cup matzoh cake meal (see Cooks' notes)", 'Kosher or fine salt', 'Freshly ground black pepper', '1 (6- to 6 1/2-pound) fi
rst- or second-cut beef brisket', '3 to 4 tablespoons vegetable oil', '16 medium shallots (about 1 pound); peeled, leaving root
ends intact', '3 large garlic cloves, finely ...
------------------------------------------------------------
Instructions:
```

```
Heat oven to 350°F with rack in middle.
Whisk together matzoh meal with 1 tablespoon kosher salt (2 teaspoons fine) and 1/2 teaspoon pepper. Pat brisket dry and dredge in matzoh mixture, shaking off excess.
Set roasting pan across 2 burners and in it heat 3 tablespoons oil over medium-high heat until oil shimmers. Brown brisket (fat side down first if using first cut) on both sides, 3 to 5 minutes...
================================================================================

================================================
TEST IMAGE 8.jpg
================================================
Recherche de 3 recettes similaires...
3 recipes similaires trouvées
```



```
================================================================================
RECETTES SIMILAIRES
================================================================================

RANG #1: 0.5943
Wine-Braised Brisket with Tart Cherries
------------------------------------------------------------
Ingredients:
["1/4 cup matzoh cake meal (see Cooks' notes)", 'Kosher or fine salt', 'Freshly ground black pepper', '1 (6- to 6 1/2-pound) fi
rst- or second-cut beef brisket', '3 to 4 tablespoons vegetable oil', '16 medium shallots (about 1 pound); peeled, leaving root
ends intact', '3 large garlic cloves, finely ...
------------------------------------------------------------
Instructions:
Heat oven to 350°F with rack in middle.
Whisk together matzoh meal with 1 tablespoon kosher salt (2 teaspoons fine) and 1/2 teaspoon pepper. Pat brisket dry and dredge
in matzoh mixture, shaking off excess.
Set roasting pan across 2 burners and in it heat 3 tablespoons oil over medium-high heat until oil shimmers. Brown brisket (fat
side down first if using first cut) on both sides, 3 to 5 minutes...
================================================================================

RANG #2: 0.5848
Yum Yum Nut Sweets
------------------------------------------------------------
Ingredients:
['1/2 teaspoon safflower oil', '1 cup whole roasted unsalted almonds', '1 cup whole walnuts', '1/3 cup sugar', '2 egg whites']
------------------------------------------------------------
Instructions:
Preheat the oven to 300°F. Grease a 7- or 8-inch square baking pan with the oil. Set aside. Place the nuts in a food processor
and chop until the mixture resembles coarse sand. Put the contents into a small mixing bowl. Stir in the sugar and egg whites.
With a spatula, scrape the contents of the bowl into the oiled pan and press into an even layer. Bake for 1 hour for soft cooki
es or 1 1/2 hours f...
================================================================================

RANG #3: 0.5762
Banana Bread
------------------------------------------------------------
Ingredients:
['1 stick butter, melted, plus softened butter for greasing', '2 cups flour', '1/2 teaspoon salt', '1 1/2 teaspoons baking powd
er', '1 cup sugar', '3 very ripe bananas, mashed with a fork until smooth', '2 eggs', '1 teaspoon vanilla extract', '1/2 cup ch
opped walnuts (optional)', '1/2 cup shredded u...
------------------------------------------------------------
```

Instructions:
Heat the oven to 350°F. Grease a 9x5-inch loaf pan with softened butter.
Whisk together the flour, salt, baking powder, and sugar in a large bowl.
Mix together the melted butter and mashed bananas in a separate bowl. Beat in the eggs and vanilla until well combined. Stir th
is mixture into the dry ingredients just enough to combine everything. Gently fold in the nuts and coconut if you’re using th
e...
==========