Tarea: DVS de una Imagen

Por: Carlos Barrera

En esta tarea se lleva a cabo la implementación del uso de la DVS de una imagen para su reconstrucción a partir de un número limitado de dichos valores con el fin de comprobar que no es necesaria la información completa de la matriz generada por la imagen para obtener una aproximación a la imagen original.

Empezamos importando las dos librerías que estaremos usando: numpy y el módulo Image de la librería PIL

In [1]:
```
import numpy as np
from PIL import Image
```

Definimos la función reconstuct_for_k, esta función recibe tres parámetros:

1. mat: La matriz que será descompuesta y luego reconstruida
2. k: el número de elementos a partir del cual se reconstruirá la imagen (puede ser el valor directo de k o un porcentaje)
3. k_type: si este parámetro tiene el valor "pctg" tomará a k como un porcentaje del número total de valores del vector s, en cualquier otro caso, la reconstrucción se hará tomando los valores 1...k más significativos del vector s. *(Nota: una ventaja de expresar k en porcentaje es que no debemos preocuparnos por la dimensión del vector s)*

In [2]:
```
def reconstruct_for_k (mat, k, k_type):
    U, s, V = np.linalg.svd(mat, full_matrices=True)
    k = get_k(s, k, k_type)
    result = np.zeros([mat.shape[0], mat.shape[1]])
    for x in range (0,k):
        result += s[x]*np.dot(np.matrix(U[:,x],float).T,np.matrix(V
[x], float))
    result = sanitize_matrix(result)
    return {'result_matrix':result, 'k': k}
```

La función reconstruct_for_k_dot recibe los mismos parámetros que la anterior y su comportamiento es exactamente el mismo;la diferencia radica en el modo de ejecutar la reconstrucción de la matriz. En el caso de la función reconstruct_for*k, se hace la suma de los productos $ u*{i}s*{i}v*{i}^{t} $ para *i* desde 1 hasta k. En el caso de esta función todos los elementos $s*i$ tales que $i>k$ se hacen cero y se aplica la función dot lo cual genera la suma deseada. $s*{1}u*{1}v*{1}^T + s*{2}u*{2}v*{2}^T + ... + s*{k}u*{k}v*{k}^T$.

Al hacer varias pruebas descubrí que generar la suma utilizando la función dot es más rápido que generar la suma elemento a elemento. En todo caso, las dos funciones son idénticas en cuanto a su uso.

In [3]:
```
def reconstruct_for_k_dot (mat, k, k_type):
    U, s, V = np.linalg.svd(mat, full_matrices=True)
    S = np.zeros([mat.shape[0], mat.shape[1]])
    k = get_k(s, k, k_type)
    S[:k, :k] = np.diag(s[:k])
    result = np.dot(U, np.dot(S, V))
    result = sanitize_matrix(result)
    return {'result_matrix':result, 'k': k}
```

La funcion get_k recibe tres parámetros:

1. s: El vector s devuelto por la función np.linalg.svd
2. k: Tiene el mismo uso y significado que en la función reconstruct_for_k
3. k_type: Tiene el mismo uso y significado que en la función reconstruct_for_k

Si k_type="pctg" esta función transforma k en el valor adecuado con respecto al número de valores s_i no nulos Esta función también limita el valor máximo de k que nunca puede ser mayor al número de valores s_i no nulos

In [4]:
```
def get_k (s, k, k_type):
    if(k_type!="pctg"):
        if(k > s.shape[0]):
            k = s.shape[0]
    else:
        if(k>=100):
            k=s.shape[0]
        else:
            k=int((k/100)*s.shape[0])

    return k
```

La función sanitize_matrix se utiliza para evitar errores de overflow cuando la matriz reconstruida se convierte a una matriz con dtype=uint8. Si no se sanitiza esta matriz los valores que provocan overflow con el tipo de dato uint8 se convierten en su complemento a 255; por ejemplo, un valor de 260 se interpretaría como 5 lo cual convertiría un pixel muy luminoso en uno oscuro; lo inverso sucede para valores menores a 0.

In [5]:
```
def sanitize_matrix (mat):
    for x in range (0, mat.shape[0]):
        for y in range (0, mat.shape[1]):
            if mat[x][y] > 255 :
                mat[x][y] = 255
            if mat[x][y] < 0:
                mat[x][y] = 0
    return mat
```

La función compress_image recibe 5 parámetros:

1. filename: El nombre del archivo de imagen a leer (para esta tarea solo se tomó en cuenta imágenes jpg, el nombre no debe incluir la extensión del archivo)
2. color_type: Si este parámetro tiene un valor de "RGB", la DVS ocurrirá en los tres canales de color resultando en una imagen a color, en cualquier otro caso, se generará una imagen en blanco y negro
3. k: Tiene el mismo uso y significado que en la función reconstruct_for_k
4. k_type: Tiene el mismo uso y significado que en la función reconstruct_for_k
5. sum_mode: Si este parámetro tiene un valor de "dot" utilizará la función reconstruct_for_k_dot para reconstruir la imagen, en caso contrario, utilizará la función reconstruct_for_k (ambas funciones son idénticas en funcionamiento)

Esta función nos regresa un diccionario con dos elementos:

1. image: Un objeto imagen generado a partir de la reconstrucción para k puntos
2. k: el número de elementos del vector s que se utilizaron para la reconstrucción

In [6]:
```
def compress_image (filename, color_type, k, k_mode, sum_mode):
    if(color_type !="RGB"):
        color_type = "L"
    img = Image.open(filename)
    img_conv = img.convert(color_type)
    img_matrix = np.array(img_conv)
    if(color_type=="RGB"):
        result = np.zeros([img_matrix.shape[0], img_matrix.shape
[1], 3])
        for x in range (0,3):
            if(sum_mode != 'dot'):
                result_obj = reconstruct_for_k(img_matrix[:,:,x],
k, k_mode)
            else:
                result_obj = reconstruct_for_k_dot(img_matrix[:,:,
x], k, k_mode)
            result[:,:,x] = result_obj['result_matrix']
    else:
        if(sum_mode != 'dot'):
            result_obj = reconstruct_for_k(img_matrix, k, k_mode)
        else:
            result_obj = reconstruct_for_k_dot(img_matrix, k, k_mod
e)
        result = result_obj['result_matrix']
    img_result = Image.fromarray(np.uint8(result), color_type)
    return {'image':img_result, 'k': result_obj['k']}
```

Uso:

Las siguientes líneas definen los valores de los parámetros para generar 10 versiones a color de la misma imagen utilizando el 10%, 20%, ..., 90% y 100% de los valores del vector s respectivamente utilizando la función dot para reconstruir la imagen.

In [7]:
```
k_options = (100, 90, 80, 70, 60, 50, 40, 30, 20, 10)
file_name = "Ygritte"
color_type = "RGB"
k_mode = "pctg"
sum_mode = "dot"
for k in k_options:
    img = compress_image(file_name+".jpg", color_type, k, k_mode, s
um_mode)
    if k_mode != "pctg":
        img_name = file_name+"_"+color_type+"_k_"+str(img['k'])+".j
pg"
    else:
        img_name = file_name+"_"+color_type+"_k_"+str(img['k'])+"_p
_"+str(k)+".jpg"
    img['image'].save(img_name, "JPEG")
```

Resultados

A continuación se mostrarán las imágenes correspondientes a los diferentes valores de k

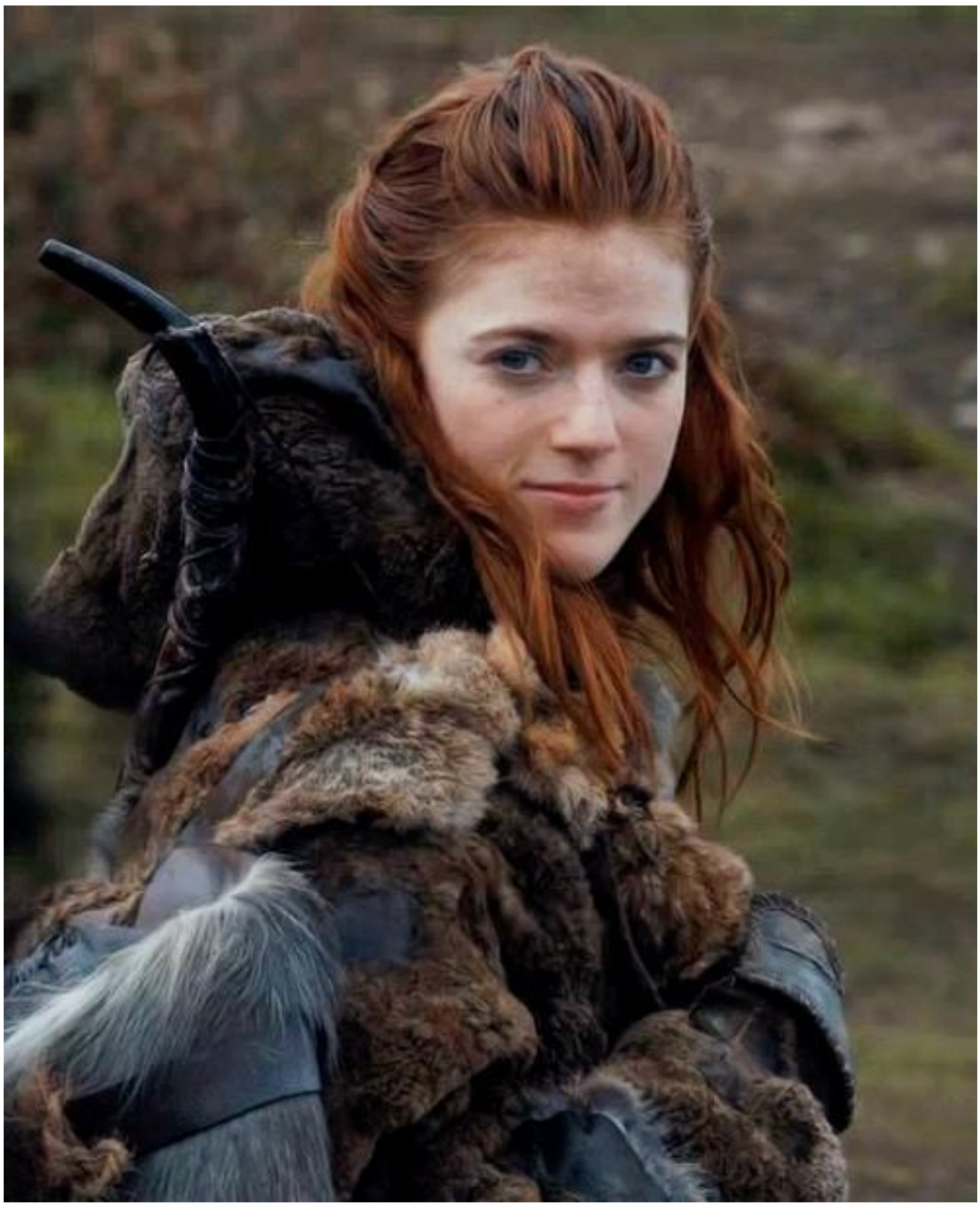
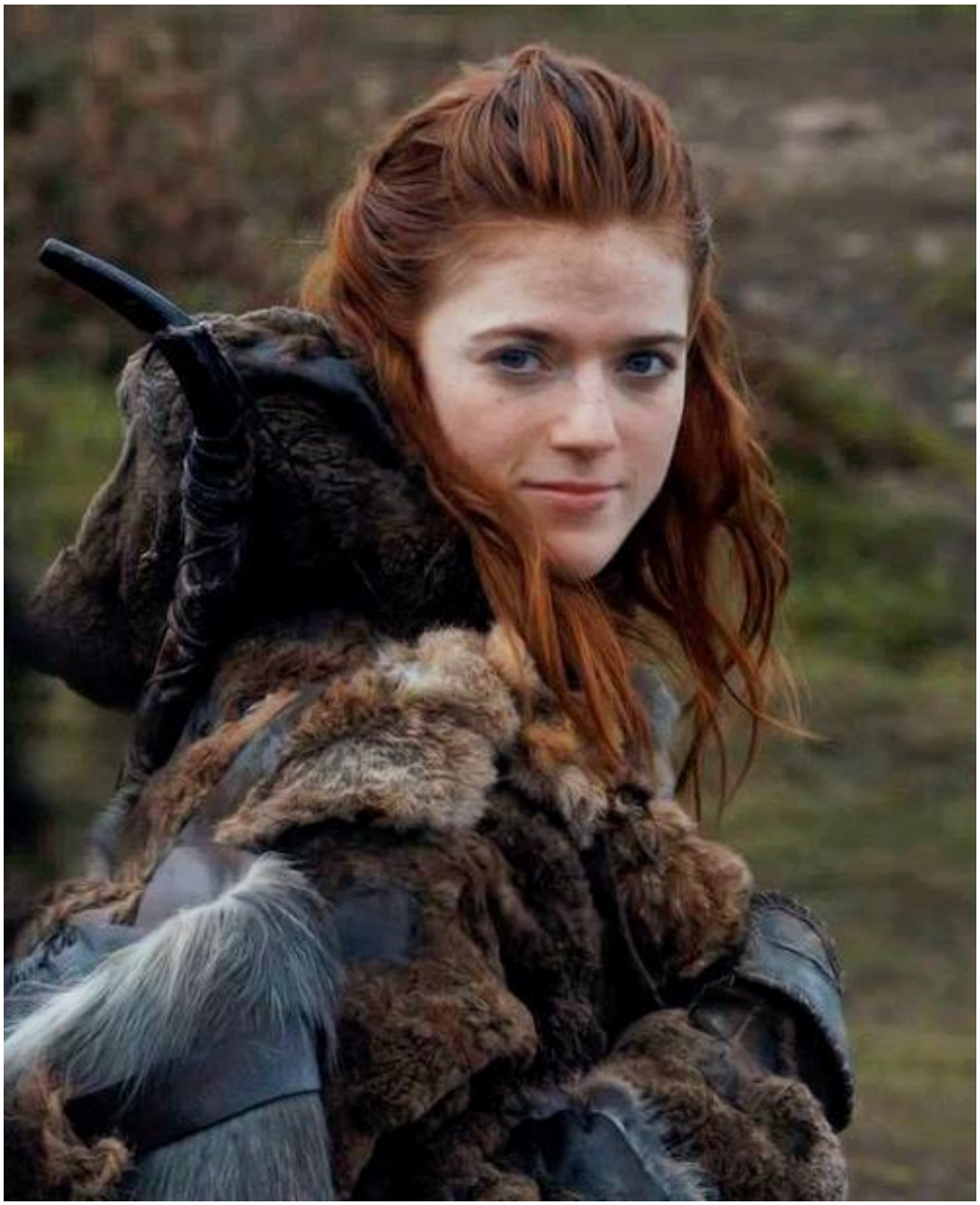
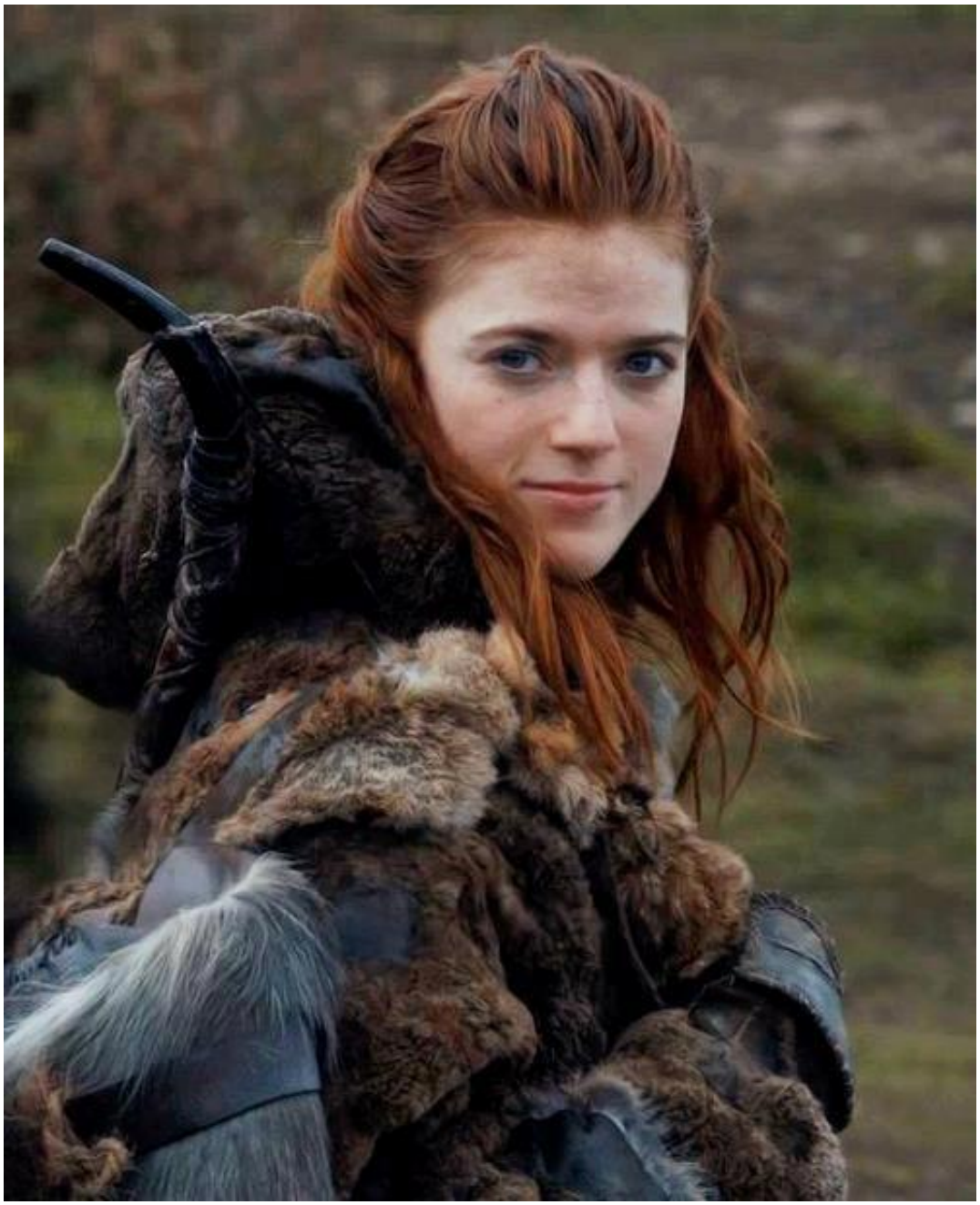


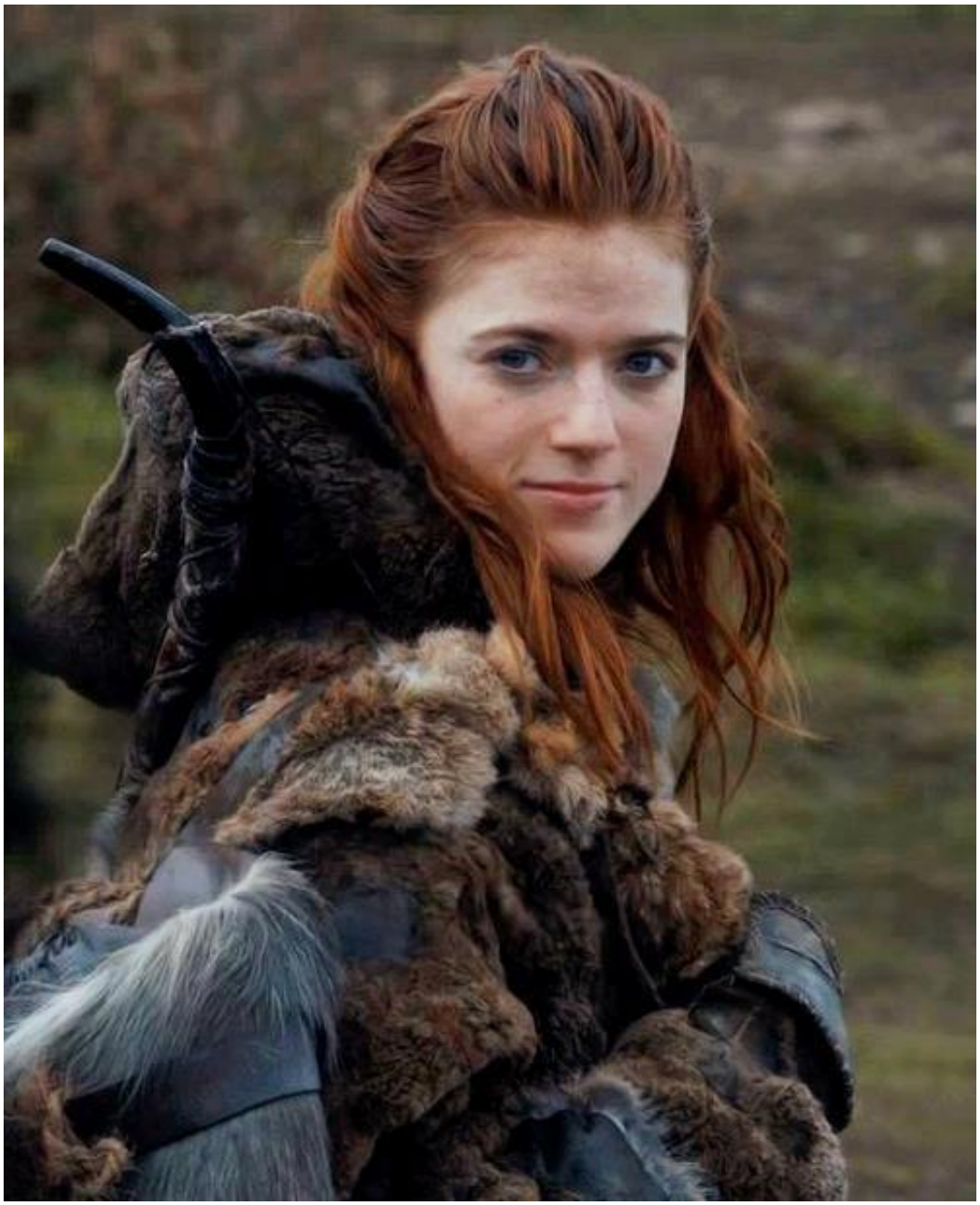
Imagen Original



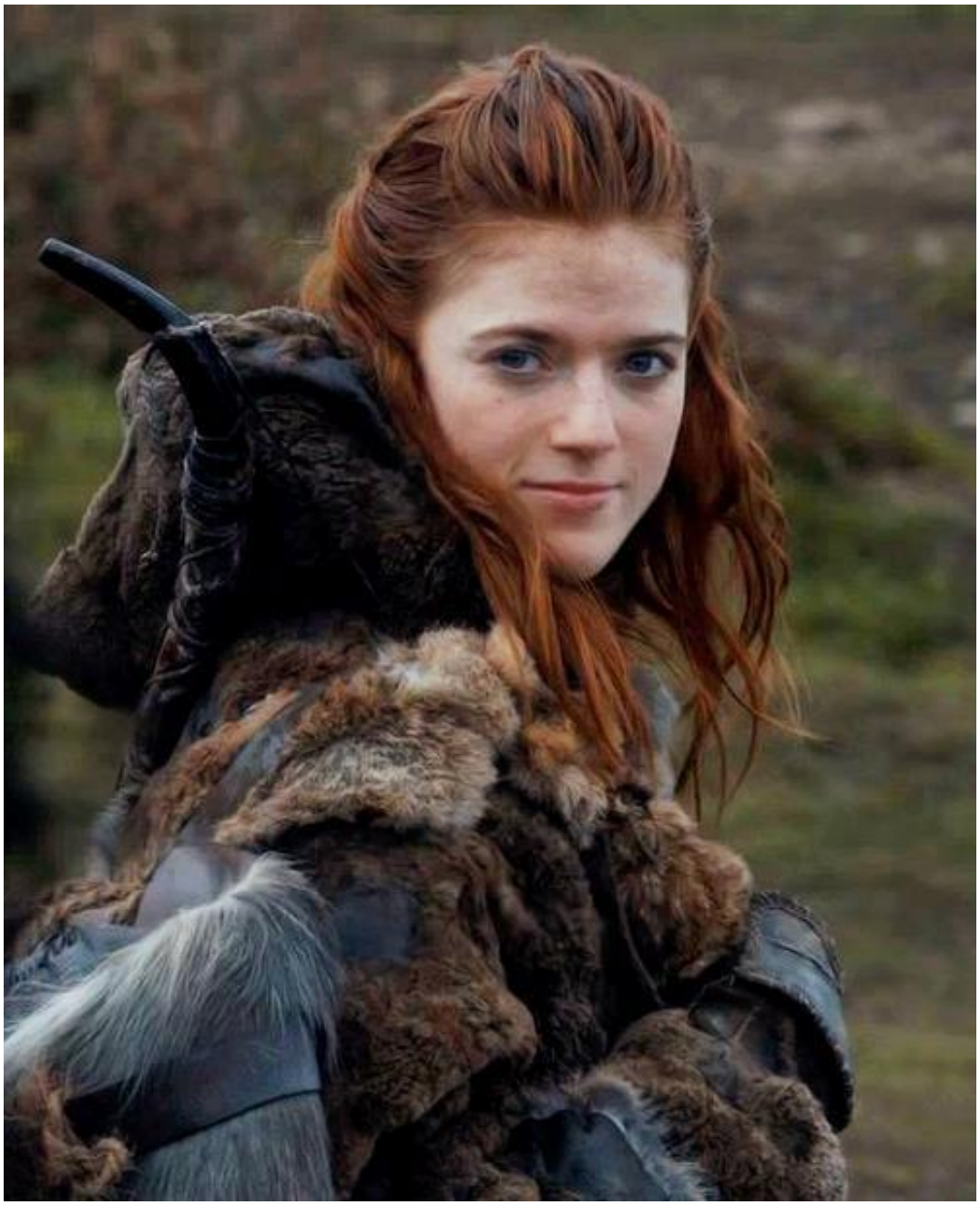
100% de los valores de s
k=500



90% de los valores de s
k=450

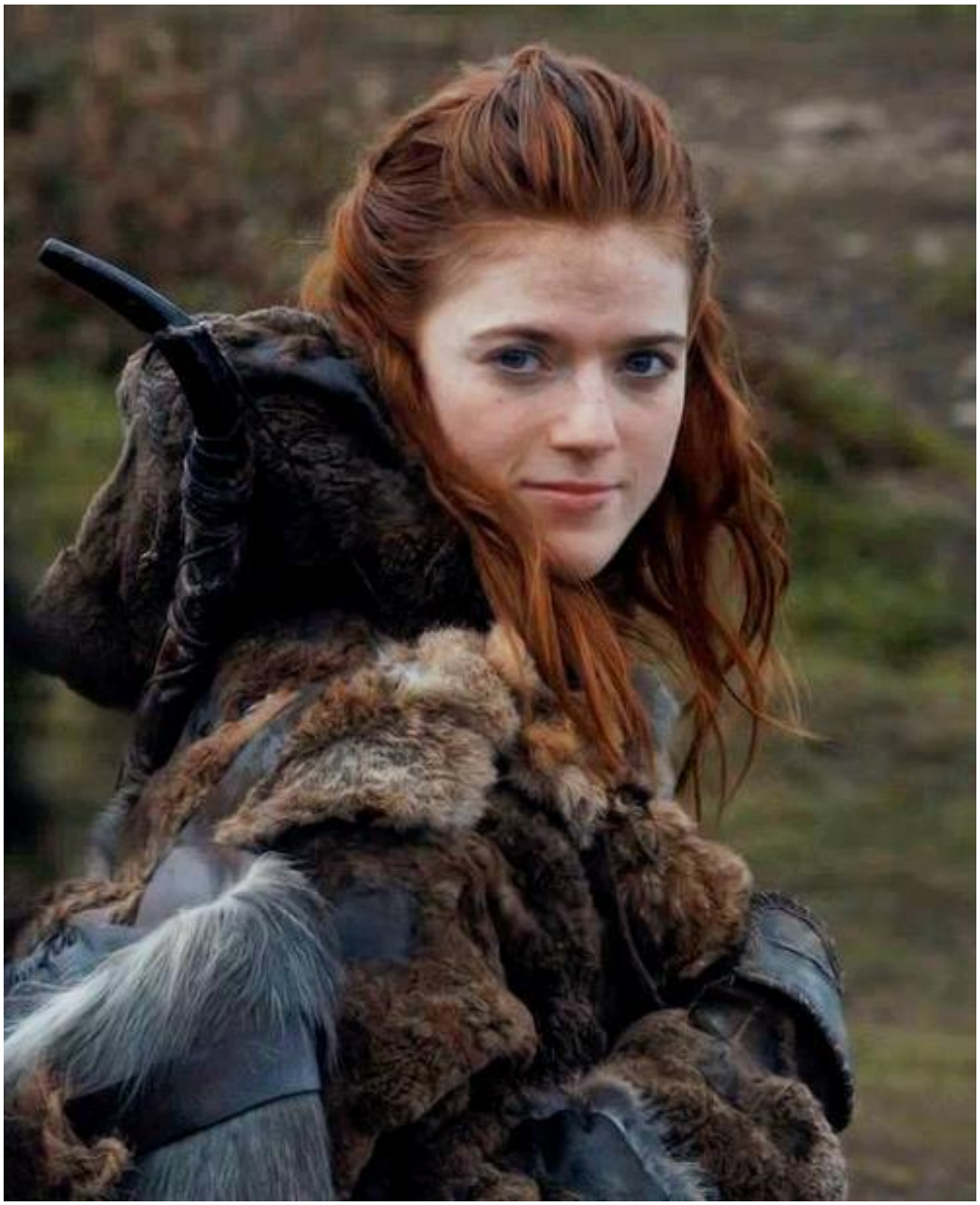


80% de los valores de s
k=400



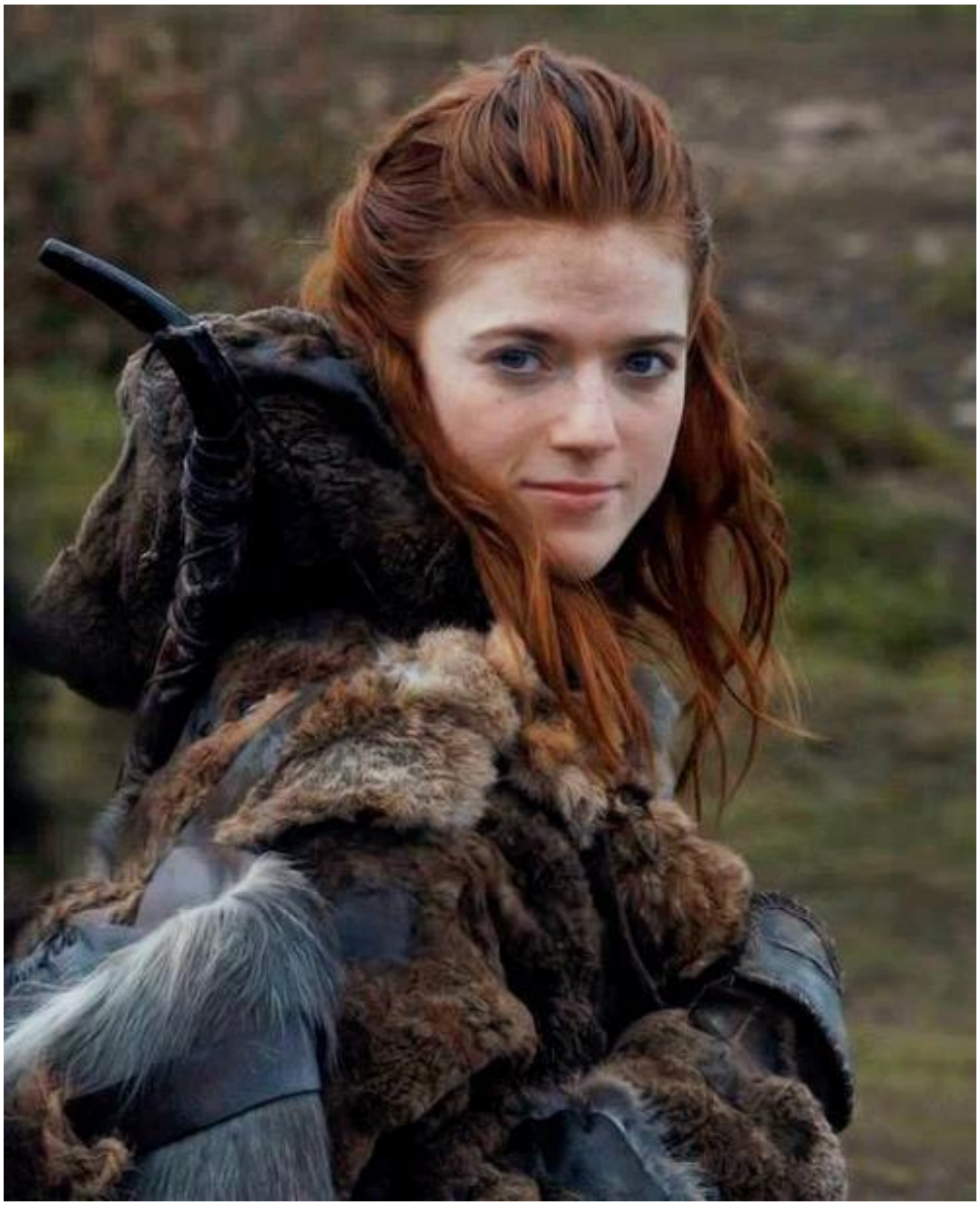
70% de los valores de s

k=350



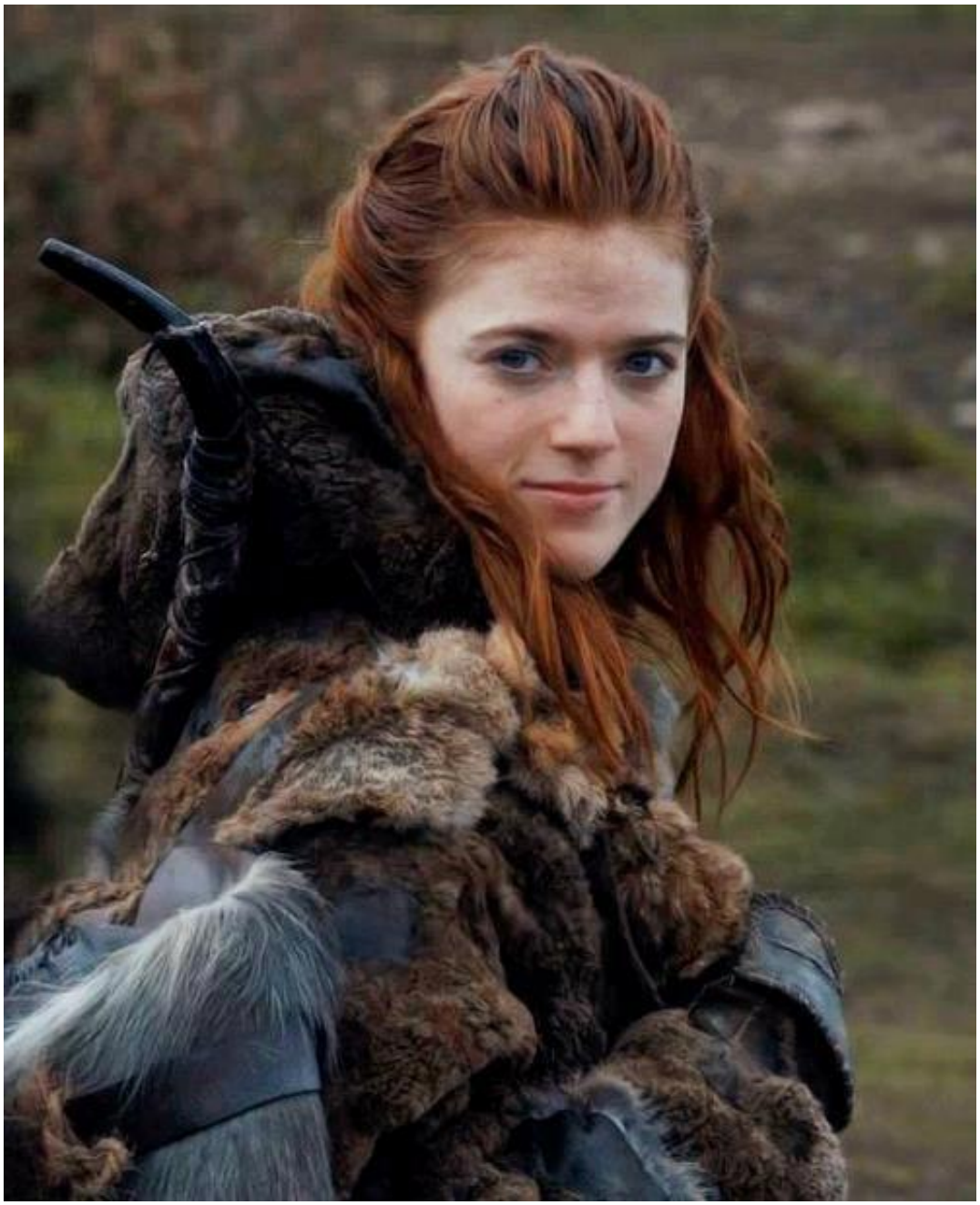
60% de los valores de s

k=300



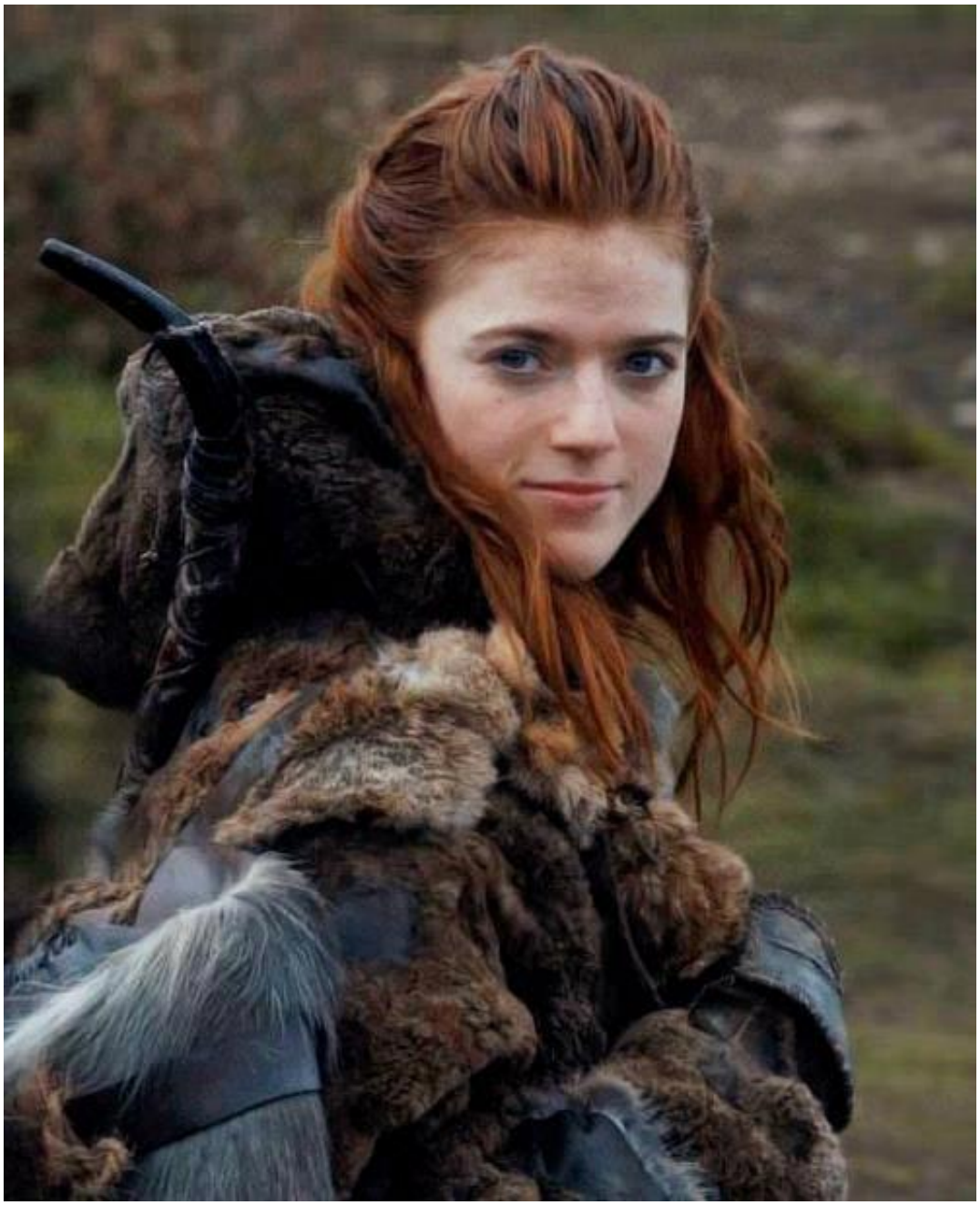
50% de los valores de s

k=250



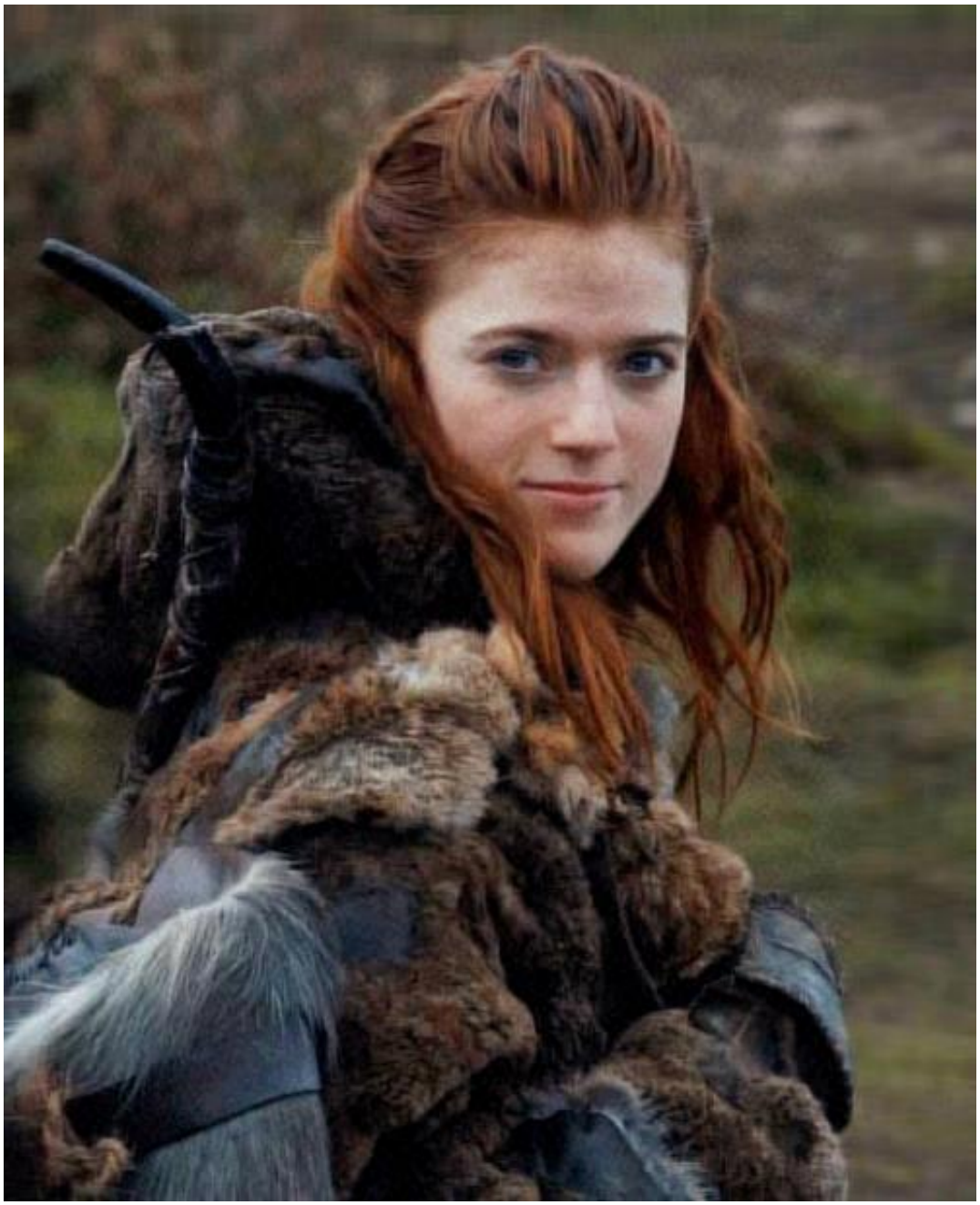
40% de los valores de s

k=200



30% de los valores de s

k=150



20% de los valores de s

k=100

10% de los valores de s

k=50

Uso 2

Las siguientes líneas definen los valores de los parámetros para generar 2 versiones a blanco y negro de la misma imagen utilizando el valores puntuales de k generando los elementos $s_{i}u_{i}v_{i}^{T}$ uno por uno para reconstruir la imagen.

In [8]:
```
k_options = (25, 50)
file_name = "Ygritte"
color_type = "L"
k_mode = "exact"
sum_mode = "sum"
for k in k_options:
    img = compress_image(file_name+".jpg", color_type, k, k_mode, s
um_mode)
    if k_mode != "pctg":
        img_name = file_name+"_"+color_type+"_k_"+str(img['k'])+".j
pg"
    else:
        img_name = file_name+"_"+color_type+"_k_"+str(img['k'])+"_p
_"+str(k)+".jpg"
    img['image'].save(img_name, "JPEG")
```

Resultados

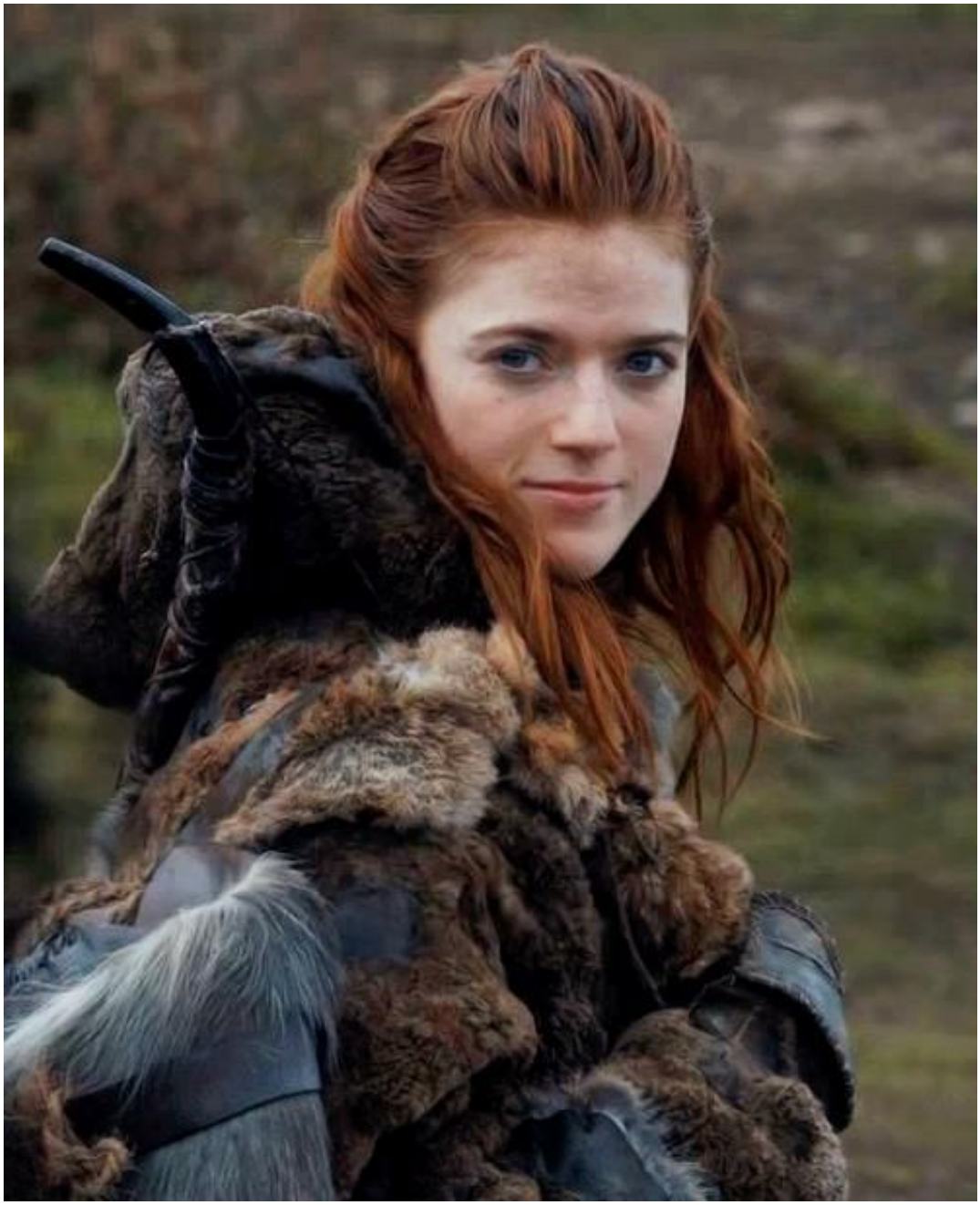


Imagen Original

k=50

(10% de los valores de s)

k=25

(5% de los valores de s)

Conclusión

Como podemos observar, es necesario utilizar valores muy pequeños de k (en este caso un número menor al 20% de los elementos del vector s) para que el cambio en la imagen sea realmente perceptible, lo cual nos demuestra que podemos reconstruir una imagen muy aproximada a la original utilizando una cantidad de información drásticamente menor.

In [ ]: