Lass uns nun ein **echtes, praktisches Beispiel für eine 4D-Tensor-Operation** in C zeigen .

🎯 Beispiel: Ein 4D-Tensor – ein **Video als 4D-Tensor**

🔍 Was ist ein 4D-Tensor?

Ein **4D-Tensor** hat **4 Dimensionen**.

Im Alltag ist das oft ein **Video**.

Ein Video besteht aus:

- **Frames** (z. B. 100 Frames = 100 Bilder)

- **Breite** (z. B. 640 Pixel)

- **Höhe** (z. B. 480 Pixel)

- **Farbkanäle** (z. B. Rot, Grün, Blau → 3 Kanäle)

➡ Das macht ein **4D-Tensor** von Größe:

`[Frames][Breite][Höhe][Kanal]`

> Das ist ein **4D-Array** – wie ein **Würfel mit 4 Dimensionen**.

📌 Beispiel: Ein kleines Video (1×640×480×3)

- 1 Frame (für die Einfachheit)

- 640 Pixel breit

- 480 Pixel hoch

- 3 Farbkanäle (R, G, B)

Wir können **einfache Operationen** durchführen – z. B. **Zusammenrechnen der Farbwerte** oder **Farbe ändern**.

✅ C-Code: 4D-Tensor – einfach, klar, für Windows

```
#include <stdio.h>


// Funktion: Ausgabe eines kleinen 4D-Tensors (1×2×2×3)
void print_4d_tensor(float tensor[1][2][2][3]) {
   printf("4D-Tensor (1×2×2×3) - Farb-Werte (R, G, B)\n");
   printf("Frame 0:\n");
   printf("  Position (0,0):\n");
   printf("    R: %.1f, G: %.1f, B: %.1f\n",
        tensor[0][0][0][0], tensor[0][0][0][1], tensor[0][0][0][2]);
   printf("  Position (0,1):\n");
   printf("R: %.1f, G: %.1f, B: %.1f\n", tensor[0][0][1][0], tensor[0][0][1][1], tensor[0][0][1][2]);
```

```
    printf("Position (1,0):\n");
    printf("R: %.1f, G: %.1f, B: %.1f\n",tensor[0][1][0][0], tensor[0][1][0][1], tensor[0][1][0][2]);
    printf("Position (1,1):\n");
    printf("R: %.1f, G: %.1f, B: %.1f\n",tensor[0][1][1][0], tensor[0][1][1][1], tensor[0][1][1][2]);
}
// Funktion: Setze einen Wert in einem 4D-Tensor
void set_value(float tensor[1][2][2][3], int frame, int x, int y, int channel, float value) {
    tensor[frame][x][y][channel] = value;
}
// Funktion: Berechne Durchschnitt der R-Werte in einem Frame
float avg_red(float tensor[1][2][2][3]) {
    float sum = 0.0;
    for (int x = 0; x < 2; x++) {
        for (int y = 0; y < 2; y++) {
            sum += tensor[0][x][y][0]; // R-Wert (Kanal 0)
        }
    }
    return sum / 4.0;
}

// Funktion: Setze alle B-Werte auf 0 (weisses Bild)
void set_blue_to_zero(float tensor[1][2][2][3]) {
    for (int x = 0; x < 2; x++) {
        for (int y = 0; y < 2; y++) {
            tensor[0][x][y][2] = 0.0; // B-Kanal wird auf 0 gesetzt
        }
    }
}
int main() {
    // Erstelle ein 4D-Tensor: 1 Frame × 2×2 Pixel × 3 Farbkanäle
    float video[1][2][2][3]; // [Frame][X][Y][Kanal
 // Fülle mit Beispielwerten (z. B. ein rotes Bild)
```

```
    set_value(video, 0, 0, 0, 0, 255.0); // R = 255 (rot)
    set_value(video, 0, 0, 0, 1, 0.0);   // G = 0
    set_value(video, 0, 0, 0, 2, 0.0);   // B = 0
    set_value(video, 0, 0, 1, 0, 100.0); // R = 100
    set_value(video, 0, 0, 1, 1, 0.0);
    set_value(video, 0, 0, 1, 2, 0.0);
    set_value(video, 0, 1, 0, 0, 150.0); // R = 150
    set_value(video, 0, 1, 0, 1, 0.0);
    set_value(video, 0, 1, 0, 2, 0.0);
    set_value(video, 0, 1, 1, 0, 50.0);  // R = 50
    set_value(video, 0, 1, 1, 1, 0.0);
    set_value(video, 0, 1, 1, 2, 0.0);

    // Ausgabe des 4D-Tensors
    print_4d_tensor(video);
    // Berechne Durchschnitt der R-Werte
    float avg_red_val = avg_red(video);
    printf("\nDurchschnittlicher R-Wert: %.1f\n", avg_red_val);
    // Setze alle B-Werte auf 0 (blau wird weg)
    set_blue_to_zero(video);
    printf("\nNachdem alle B-Werte auf 0 gesetzt wurden:\n");
    printf("  Position (0,0,0): R=%.1f, G=%.1f, B=%.1f\n",
        video[0][0][0][0], video[0][0][0][1], video[0][0][0][2]);
    printf("  Position (1,1,1): R=%.1f, G=%.1f, B=%.1f\n",
        video[0][1][1][0], video[0][1][1][1], video[0][1][1][2]);
    return 0;
}
```

🔍 Was passiert hier?

| Schritt | Was tun |

|--------|--------|

| 1 | Erstelle ein 4D-Array: `float[1][2][2][3]` |

| 2 | Setze Farbwerte für einzelne Pixel (R, G, B) |

| 3 | Zeige die Werte in einer übersichtlichen Tabelle |

| 4 | Berechne Durchschnitt der Rot-Werte (R) |

| 5 | Setze alle B-Werte auf 0 → das Bild wird **blau frei** |

📝 Ausgabe auf der Konsole:

4D-Tensor (1×2×2×3) - Farb-Werte (R, G, B)

Frame 0:

Position (0,0):

R: 255.0, G: 0.0, B: 0.0

Position (0,1):

R: 100.0, G: 0.0, B: 0.0

Position (1,0):

R: 150.0, G: 0.0, B: 0.0

Position (1,1):

R: 50.0, G: 0.0, B: 0.0

Durchschnittlicher R-Wert: 112.5

Nachdem alle B-Werte auf 0 gesetzt wurden:

Position (0,0,0): R=255.0, G=0.0, B=0.0

Position (1,1,1): R=50.0, G=0.0, B=0.0

🚀 Warum ist das wichtig?

- In **Künstlicher Intelligenz** (z. B. CNN): Bilder werden als 4D-Tensoren (Frame × H × W × 3) verarbeitet.

- In **Computer-Vision**: Jedes Bild ist ein 4D-Tensor.

- In **C-Programmen**: Du kannst diese Daten direkt verarbeiten – **ohne Python, ohne BLAS, ohne Linux**.

📌 Kurz: Was ist ein 4D-Tensor?

> Ein **4D-Tensor** ist ein **mehrdimensionaler Kasten** mit 4 Dimensionen.

> Es ist wie ein **Video**, das aus **Frames, Breite, Höhe und Farbkanälen** besteht.

> In C wird es als **4D-Array** dargestellt: `float[1][2][2][3]`.

✅ Fazit

✅ Dieses Beispiel zeigt:

- Ein **echtes, realitätsnahes 4D-Tensor-Beispiel** (ein Bild als Video-Fragment)

- Wie man es **in C erzeugt, ausgibt und verarbeitet**

🚀 Nächste Schritte?

(Einfach weiter die KI Fragen mit diesen Fragen

Wenn du möchtest, kann ich dir auch zeigen:

- Wie man **2 4D-Tensoren addiert** (z. B. zwei Bilder zusammen)

- Wie man ein **3D-Tensor in einen 4D-Tensor umwandelt**

- Oder wie man **Tensoren in einem Deep-Learning-Modell simuliert**